COMPUTER AIDED VERIFICATION OF LOGICAL DESIGN OF PROCESS CONTROL RELAY SWITCHING CIRCUITS

by

P C DIXIT

COMPUTER SCIENCE PROGRAMME

TUTE OF TECHNOLOGY KANPUR

JULY 1976

COMPUTER AIDED VERIFICATION OF LOGICAL DESIGN OF PROCESS CONTROL RELAY SWITCHING CIRCUITS

A Thesis Submitted
In Partial Fulfilment of the Requirements
for the Degree of

MASTER OF TECHNOLOGY

by

P C DIXIT

to the

COMPUTER SCIENCE PROGRAMME
INDIAN INSTITUTE OF TECHNOLOGY KANPUR
JULY 1976

CSP-1979-M-DIX-COM

CERTIFICATE

This is to certify that the thesis entitled, COMPUTER AIDED VERIFICATION OF LOGICAL DESIGN OF PROCESS CONTROL RELAY SWITCHING CIRCUITS is a record of the work carried out under my supervision and that it has not been submitted elsewhere for a degree

Kanpur
July 1976

V Rajaraman
Professor and Convener
Computer Science Program
INDIAN INSTITUTE OF TECHNOLOGY, KANPUR

ACKNOWLEDGEMENTS

I am deeply indebted to Professor V Rajaraman for his continued guidance, supervision and encouragement throughout this work

I would like to express my gratitude to the Power Projects Engineering Division, Department of Atomic Energy, Bombay for sponsoring me for the studies here I am particularly grateful to Sarvasri B S Prabhakar and M M L Gulati for their appreciation of the present work and their encouragements

I am also very grateful to Dr R M K Sinha, Sarvasri S Kapoor, M S Krishnamoorthy, V M Mehrotra, Lieut N C Sarangi, Sri M H Shah and Sri Sudhir K Shukla for their help and cooperation.

Thanks are also due to Sri H K Nathani for an excellent job in typing and to Sri H S Tewari for a fine job in cyclostyling

Finally, I wish to thank my wife, Bindu, for her help and understanding

P C Dixit

- P C Dixit

Kanpur
July 1976

CONTENTS

ABSTRACT

Whether a given relay logic circuit meets the desired operating sequence can be verified by constructing a table in which status of all the contacts and relay coils is shown against successive input conditions and the status of outputs thus derived from the status of contacts However, in the case of large and complex relay logic circuits, with added complexity due to the considerations of the time-delays associated with the operation of some of the relays, the task of constructing such tables is extremely tedious and prone to error

As an alternative method a computer program has been developed for analysing relay switching circuits used in Process Control applications, with the following features

(1) It requires very little user effort in coding the relay logic circuit

(2) It provides diagnostic aid in case of any error in the design of relay circuit is detected.

(3) It provides easily digestable print-outs in the form of tables, explanatory messages, etc.

CHAPTER 1

INTRODUCTION

Although there is an increasing use of computers, particularly mini-computers, in all the areas of industrial control, but still there is a large class of production machines and processes where these computers are not cost effective This class includes automatic tube welding machines, gear grinders, chemical batch processes, petro-chemicals, small automatic telephone exchanges, etc. To site an example, almost all the nuclear power plants built until 1970 and most of those being built currently employ electro-mechanical relays for switching Use of digital computers for on-line control of nuclear power plants is still in experimental phase

The machines and processes cited above are still commonly controlled by electro-mechanical relays Inputs to the relay system come from push-buttons, multi-position switches, relay contacts, limit switches and other process-controlled switches such as pressure switches, temperature switches, etc Outputs direct power to motors, solenoids, signal lights, etc The internal wiring of the contacts of relays embodies the desired operating sequence

Whether a given relay logic circuit meets the desired operating sequence can be verified by constructing a function table in which status of all the contacts and relay coils is shown against successive input conditions and the status of outputs thus derived from the status of contacts However, in the case of large and complex relay logic circuits having large number of relays and contacts, with added complexity due to considerations of pick-up and drop-out delays associated with some of the relays, the task of constructing such function-tables is extremely tedious, time consuming and prone to errors

Any error in the design (in this thesis word design will be used to mean logical design) of relay circuits, generally comes to notice at the time of commissioning of the system A circuit operation which is not consistent with the desired operating sequence can be due to

[a] an error in the logical design

[b] an error in the engineering design

[c] an error in the assembly (wiring of relay coils and contacts) of the circuit

[d] an error in the external connections to the input contacts or outputs.

Thus the task of determining the cause of error is aggravated by these factors The first and the most important is, of course, an error in logical design If the design of the logic circuit can be proved in advance commissioning delays can be reduced In applications such as nuclear power plants where there are a large number of such relay-logic circuits, the saving in commissioning time will be well worth the effort in proving the design in advance

Thus we see that (a) proving the design of a relay circuit before it is wired is very much useful, (b) proving the design by a function table is not always satisfactory

Accordingly it was felt that a program should be developped with the following objectives

1 - It should be able to verify the logical design of relay circuits employed in process-control applications

2 - It should require minimum of user effort in the preparation of data-cards

3 - It should provide diagnostic aid in case any error in the design of the relay circuit is detected

4 - It should provide easily digestable printouts in the form of tables, explanatory messages, etc.

The second point (2) was very important, since if the procedure for preparation of data cards were complex, the advantages of the program compared to the function table will be neutrialized by the effort to code the circuit Also, it will, then, be prone to error

A program has been developed for verifying the design of relay-circuits having a maximum of 90 elements and other specifications in Appendix - A Procedure for preparing data cards is very simple and, as such, data can be prepared by a draftsman and results presented to the designer

With this brief first chapter introducing the subject, this thesis has been arranged into following 5 chapters, and two appendices

The Chapter 2 introduces some representative relay logic circuits There we arrive at a general model which will meet most of the process control applications

Chapter 3 describes the data structure and various fields on the data cards This information is a prerequisite to Chapter 4

Chapter 4 describes in detail the algorithm employed for verification of design of a relay circuit

In Chapter 5 conclusions of the work and scope for future work are presented

Appendix A lists the specifications of the relay circuits which can be analyzed by the program

Appendix B is the User's Manual This manual explains, with examples, preparation of data cards, precautions, interpretation of print-outs, simulation of faults, etc

CHAPTER 2

RELAY LOGIC CIRCUITS

2 0 INTRODUCTION

Before we develop a procedure for the verification of the design of relay logic circuits, we must be familiar with the symbols and operation of various elements used in such circuits Following the explanation of symbols and operation of the elements, we will describe some relay circuits to illustrate the concepts Then we will describe the general model and states of a relay circuit

2 1 SYMBOLS

A host of symbols or pictures are used in the industry and literature for relays, their contacts, manually operated switches, signal lights, etc A brief description of various such elements is given below Symbols are shown in Figure 1 Equivalent symbols in any standard such as JIC (Joint Industry Conference), NARM (National Association of Relay Manufacturers), ISA (Instrument Society of America), etc , can be worked-out from these descriptions

2 1 1 Relay The relay considered in this thesis is one which has two terminals for the supply of electrical power to energize it When current flows through

these two terminals the relay picks-up, else it is in dropped-out state

Relays are of two types The no-delay relays (NDR's) are the relays which pick-up and drop-out immediately on application of power or its withdrawal The time delay relays (= TDR's) are the relays which, on application of power, pick-up only after a lapse of a fixed time called pick-up delay, and/or drop-out, on withdrawal of power, only after a lapse of a fixed time called its drop-out delay The delays associated with a TDR are written on a side of its symbol

In this thesis, relays will be labelled as R followed by an identification number, e g , R1, R105, RAX, RA, etc

2 1 2 Relay Contacts Various forms of the relay contacts are as follows

FORM A Also known as normally open (=No) contact is open when relay is in dropped-out state It is closed (= conducting) when the relay is in picked up state

FORM B Also known as normally closed (=NC) is closed when relay is in dropped-out state It is open when relay is in picked-up state Thus, form B is logical compliment of Form A.

FORM C This is a transfer contact of the break before make type It is equivalent to, and is shown as such, two contacts, as shown in Figure 1 The part which is normally open is called Part A, and the other is called Part B When the relay picks-up, first the Part B opens, then the Part A closes Thus for a brief interval, both the parts A and B are open Similarly when the relay drops, both the parts are open for a brief interval

FORM D This is similar to Form C except that it is a continuity transfer contact During the transition, both the Parts A and B are closed

Contact Labels Contacts will be labelled with their relays as pre-fix, and a serial number as suffix For example R1 - 1 is the contact number 1 of relay R1

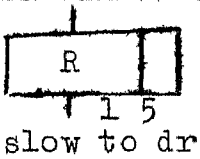
	Name of the Element	Symbol
RELAYS	Relay	
	Time Delay Relay	 slow to drop

Figure 1 (continued)

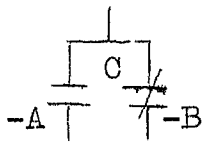
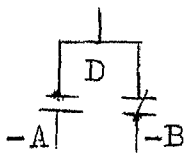
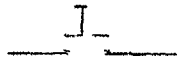
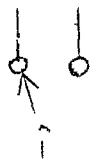
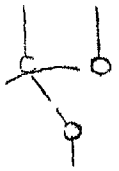
	Name of the Element	Symbol
relay contacts	Normally Open (=NO) or Form A Contact	
	Normally Closed (=NC) or Form B Contact	
	Form C Contact	
	Form D Contact	
manually operated switches	Push button, spring return, contact closed when pushed	
	Push button, spring return, contact open when pushed	
	Push button, spring return, Form C contact	
	Switch (lever), 2 position SPDT, no spring return (transfer is break before make)	
	Switch (lever), 2 position SPDT, no spring return (with a make before break transfer)	

Figure 1 (continued)

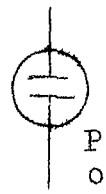
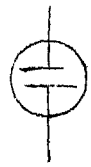
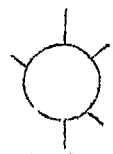
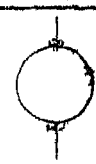
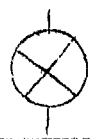
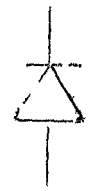
	Name of the Element	Symbol
process controlled contacts	Any process controlled contact is shown as a normally open or (arbitrarily) a closed contact enclosed in a circle, with its actual operation described by the side of the symbol	 PS-37 opens at high pressure
		 TS-1 closes at 510°C
output devices	Solenoid	 SV
	Light (Electric Bulb)	 LT
	Motor	 M
	Any other output device	
miscellaneous	Resistor	 RS
	Diode	 Cathode anode

Figure 1

2 1 3 Manually Controlled Contacts Manually controlled contacts are the contacts of push buttons, lever switches, knife switches, rotary switches, key switches, plug and jack combinations, etc Some representative ones are shown in Figure 1 They are listed and described in many catalogs and technical advertisements and no attempt will be made here to present particular descriptions or details Of interest to us is the state (open or closed) of the manually controlled contact and not the mechanism of its actuation

2 1 4 Process Control Contact These are the contacts which are actuated by the process These include pressure switch, flow switch, level switch, temperature switch, etc These will be shown, arbitrarily, as an open or closed contact, enclosed in a circle Actual operation of the contact will be written by the side of the symbol

2 1 5 Input Contacts A contact which is not a contact of any of the relays in the circuit is an input contact for the circuit These are manually controlled contacts,, process control contacts and contacts of any relays not shown in the circuit

2 1 6 Output Devices These are solenoids, electric bulbs (lights), motors, etc These are not sensitive to the direction of flow of current through them

2 1 7 Resistor In relay logic circuits, resistors are generally employed for limiting current under certain conditions

2 1 8 Diodes Diodes are sometimes employed to obtain unilateral conductance Their symbol is shown in Figure 1

2 2 EXAMPLES OF RELAY CIRCUITS

Relay switching circuits are asynchronous sequential circuits (Kohavi, Chapter 11) We will now describe some relay circuits

Example No 1

This has been taken from the book by Kohavi, Chapter 11 The circuit is shown in Figure 2

It has two input contacts X1 and X2 The output device to be controlled is a light LT Its intended operation is as follows

Initial inputs are X1=0 (i.e , open), X2=0 The LT is to be ON (=1) if and only if both X1 and X2 are 1, and the preceding input were X1=0, X2=1

If the circuit were correctly designed, on the application of the following input sequence we should

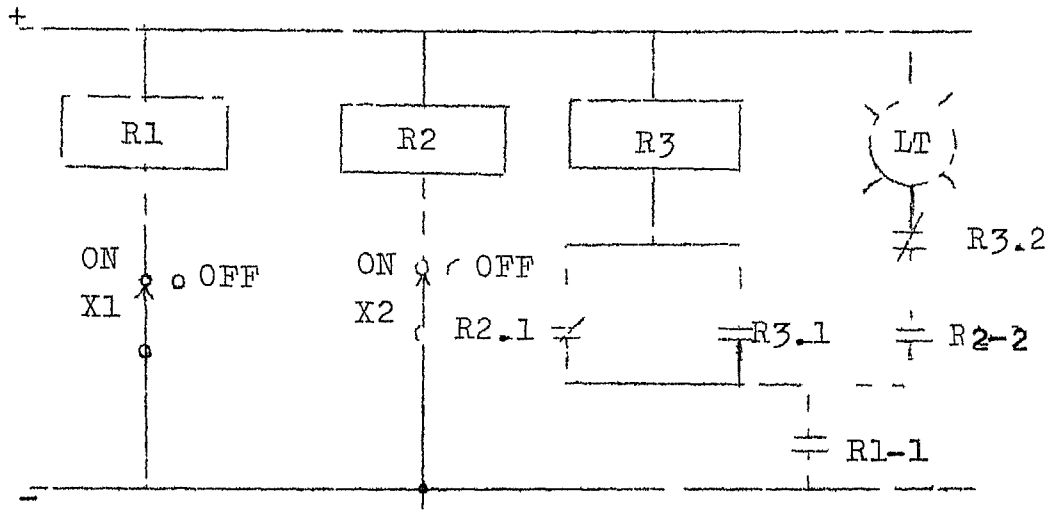


Figure 2

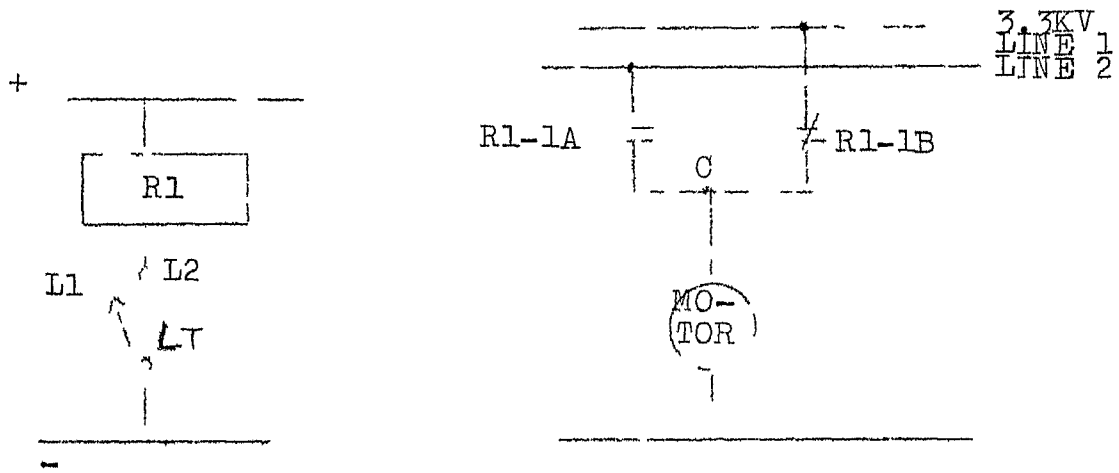


Figure 3

get the output sequence as shown against input conditions

	Inputs		Output
	X1	X2	LT
(1)	0	0	0
(2)	0	1	0
(3)	0	0	0
(4)	1	0	0
(5)	1	1	0
(6)	0	1	0
(7)	1	1	1
(8)	0	1	0

To verify the design, we will draw a function table. It is seen from the function table drawn below, that this circuit meets the specified output sequence

	Inputs		Relays and their contacts								Output
	X1	X2	R1,	R1-1,	R2,	R2-1,	R2-2,	R3,	R3-1,	R3-2	LT
(1)	0	0	0	0	0	1	0	0	0	1	0
(2)	0	1	0	0	1	0	1	0	0	1	0
(3)	0	0	0	0	0	1	0	0	0	1	0
(4)	1	0	1	1	0	1	0	1	1	0	0
(5)	1	1	1	1	1	0	1	1	1	0	0
(6)	0	1	0	0	1	0	1	0	0	1	0
(7)	1	1	1	1	1	0	1	0	0	1	1
(8)	0	1	0	1	1	0	1	0	0	1	0

Example No. 2

This example shows the special considerations that are involved in the use of transfer contacts The circuit for this example is shown in Figure 3 Its operation is as follows Under normal operating conditions the line transfer switch LTS is at position L1 and the motor gets power through the contact R1-1B. However, when some maintenance work is to be done in line LINE1, the

motor is switched to the alternative supply line LINE2 This is achieved by putting the switch LTS to position L2

If it were required that the power to the motor should not be interrupted even momentarily while switching supply from LINE1 to LINE2 and vice versa, it will be seen that this circuit will not meet the requirement This is because while transferring, momentarily both R1-1A and R1-1B are open In this case using Form D contacts will meet the requirements In some other applications form D will not meet the requirements and Form C would be used The point to remember while using a transfer contact is the transitory state (0,0) or (1,1) of these contacts

Example No 3

Circuit for this example is shown in Figure 4 This example illustrates the operation of TDR's This also illustrates the fact that in case of the circuits having TDR's, state of the circuit depends not only the sequence of inputs but may also depend on the precise moments of occurrence of the input status changes

The process requirements are that on occurrence of a trip the dump valve should open fully within 0 8 seconds Reclosure of dump valve, on clearance of the trip, should be prohibited if the dump valve had not

opened within 0 8 seconds The circuit in Figure 4 provides for the above process requirements The dump valve (not shown in the figure) is operated by SV

Under normal plant operating conditions, TRIPS=1, R2=1, R3=1 (to start with RESET is pressed momentarily which makes R3=1 Subsequently when RESET is released, R3 remains energized through its own contact R3-1)

Following occurrence of a trip (TRIPS=0), if LIMS closes before R2-1 opens (i.e , if LIMS closes within 0 8 seconds), R3 remains energized and hence R3-2 remains closed Later when TRIPS=1, the SV gets energized and dump valve closes However, on occurrence of trip if R2-1 had opened before the LIMS closed (i e , LIMS closed after 0 8 seconds) then the R3 gets de-energized and later when TRIPS=1, R3 does not pick-up since both R3-1 and RESET are 0 Thus reclosure is prohibited To re-close the dump-valve, then, the RESET is pressed when TRIPS = 1.

Thus it is seen that not only the closure but also the precise moment of closure of LIMS is important in determining the resultant state of the circuit

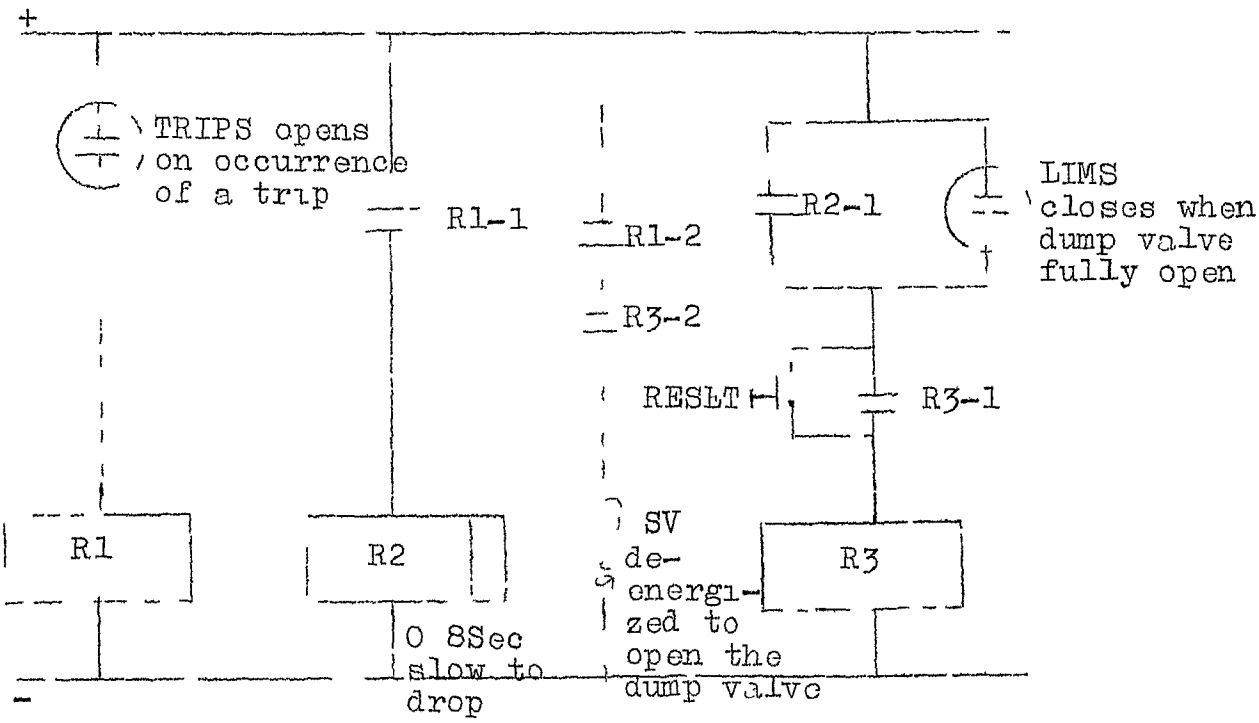


Figure 4

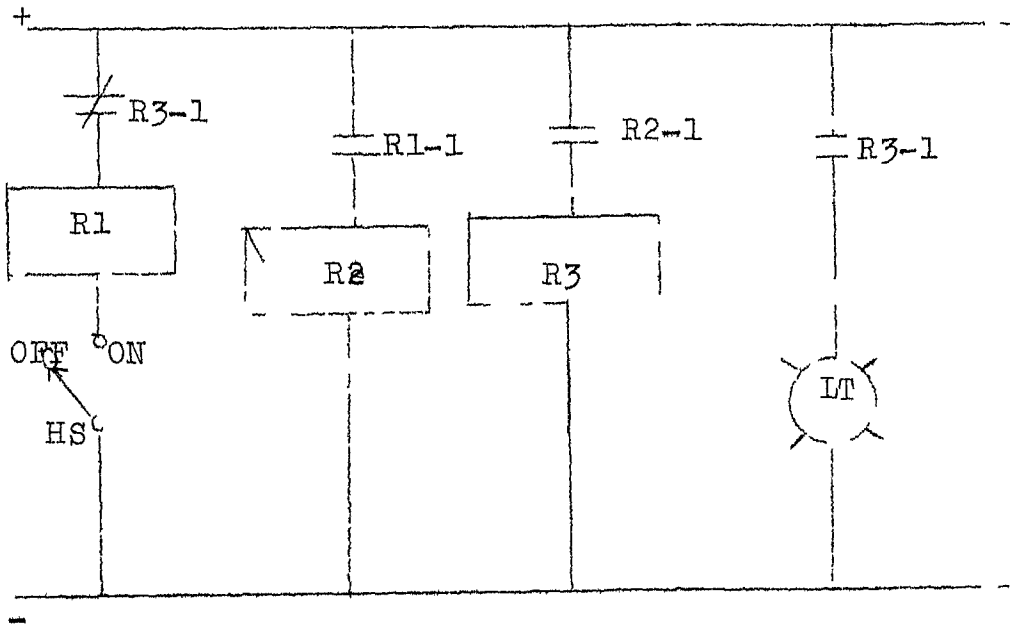


Figure 5

2 3 CYCLING

Under certain input conditions a relay circuit may exhibit cycling i e , if left with those input conditions, it never attains a steady state Instead it goes through a sequence of states repeatedly for ever One such circuit is shown in Figure 5 When the HS is in position ON, this circuit exhibits cycling In such cases, the circuit state in response to any subsequent changes in the input states, may be indeterministic. The program detects the setting in of any cycling in the circuit under any input conditions Further processing is, then, terminated with a message

2 4 RELAY CIRCUIT MODEL

The general model of the relay circuit is shown in Figure 6 The three blocks in this model have already been described in Section 2 1

2 5 STATES OF THE ELEMENTS AND THE CIRCUIT

2,5 1 States of the Elements

Contacts A contact can have only two states When it is closed (i e conducting) it is said to be in State 1. When it is open, it is said to be in State 0

Relays When a relay is in dropped out state, it is said to be in State 0 When a relay is in picked up state its state is 1

States 2 and 3 apply only to the relays having any transfer contacts These are as follows

A relay having transfer contacts is said to be in State 2 when it has been energized from its dropped out state, but its transfer contacts are in transistory phase It is said to be in State 3 when it has been de-energized from a picked-up state, but its transfer contacts are in transistory phase

If a relay is a TDR, it can have two more states which are as folllows

A TDR is said to be in State 8 when it has been energized from a dropped out state, but it is timing out its pick up delay Similarly it is said to be in State 9, when it has been de-energized from a picked up state, but it is timing its drop-out delay

The relation between the states of a relay and its contacts is shown in the Table 1

TABLE 1

State of the relay	Corresponding State of its Contacts			
	Form A	Form B	Form C -A, -B	Form D -A, -B
0	0	1	0,1	0,1
2	1	0	0,0	1,1
1	1	0	1,0	1,0
3	0	1	0,0	1,1
8	0	1	0,1	0,1
9	1	0	1,0	1,0

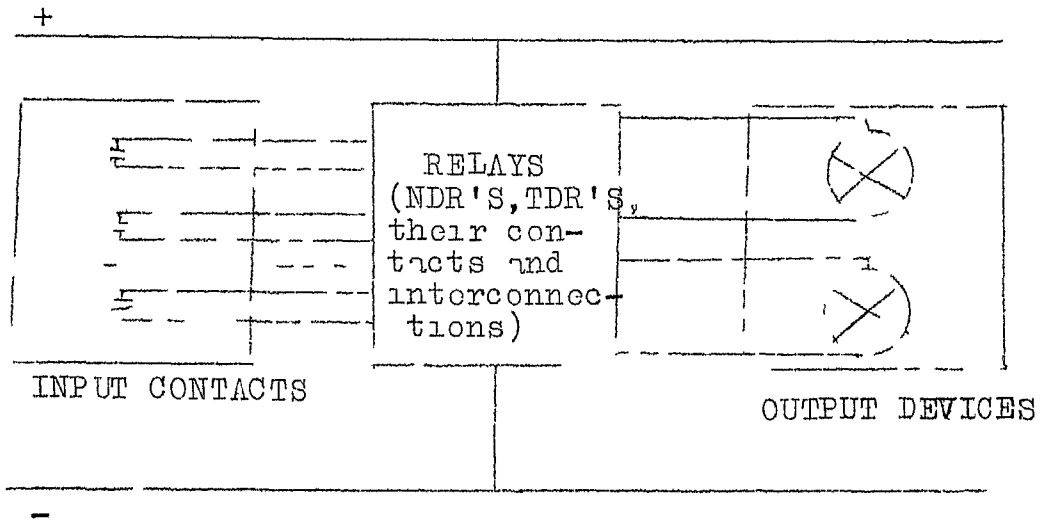


Figure6 Relay Circuit Model

Output Devices An output device is said to be in State 0 if current is not passing through it It is said to be in State 1 if current is passing through it

2 6 STATES OF A RELAY CIRCUIT

State of a relay circuit at any instant of time is defined by the states of all the elements in the circuit A circuit goes from one state to another (next) state by a change in the state of one or more of its elements A circuit is, at any instant of time, in one of the following states

2 6 1 Steady State A circuit is said to be in steady state when the state of none of its elements will change with time, unless a change in the state of one or more inputs occurs

2 6 2 Quasi Steady State This state can occur only in circuits having TDR's A circuit is said to be in quasi-steady state, if the state will change even if none of the input states change, only when some TDR times out

2 6 3 Transient State A circuit is said to be in a transient state if the state will change with time even if there is no change in the states of the inputs, and (if there are any TDR's in the circuit) even if no TDR's are in timing phase

Each transient state lasts for about 15 milliseconds Successive transient states occur for two reasons

(i) inherent delay in the operation (picking up or dropping out) of no-delay relays

(ii) transistory state (0,0 or 1,1) of transfer contacts

Although we have defined NDR's as those which operate immediately, in actual practice a small amount of time τ of the order of 15 milliseconds is required for even the no-delay relay to operate This, as will be shown by an example below, leads to transient states of the circuit

2 6.4 An Example for the States of a Relay Circuit

The circuit is shown in Figure 7 Initially TS is set to 1 (closed) and power switch is turned on at t=0 Various states through which the circuit moves to a steady state are as follows A dash (-) indicates no change in the state

Time	TS	R1	R1-1	R2	R2-1	R2-2A	R2-2B	R3	R3-1	R4	R4-1	LT	SV	STATE
0	1	0	0	0	0	0	1	0	0	0	1	1	0	Tran
τ	-	1	1	-	-	-	-	-	-	1	0	0	-	Tran
2τ	-	-	-	2	1	0	0	8	-	0	1	1	-	Tran
3τ	-	-	-	1	-	1	-	-	-	-	-	-	-	Tra
4τ	-	-	-	-	-	-	-	-	-	1	0	0	-	Quas
$2\tau+$ 0 8 sec	-	-	-	-	-	-	-	1	1	-		-	1	Ste dy.

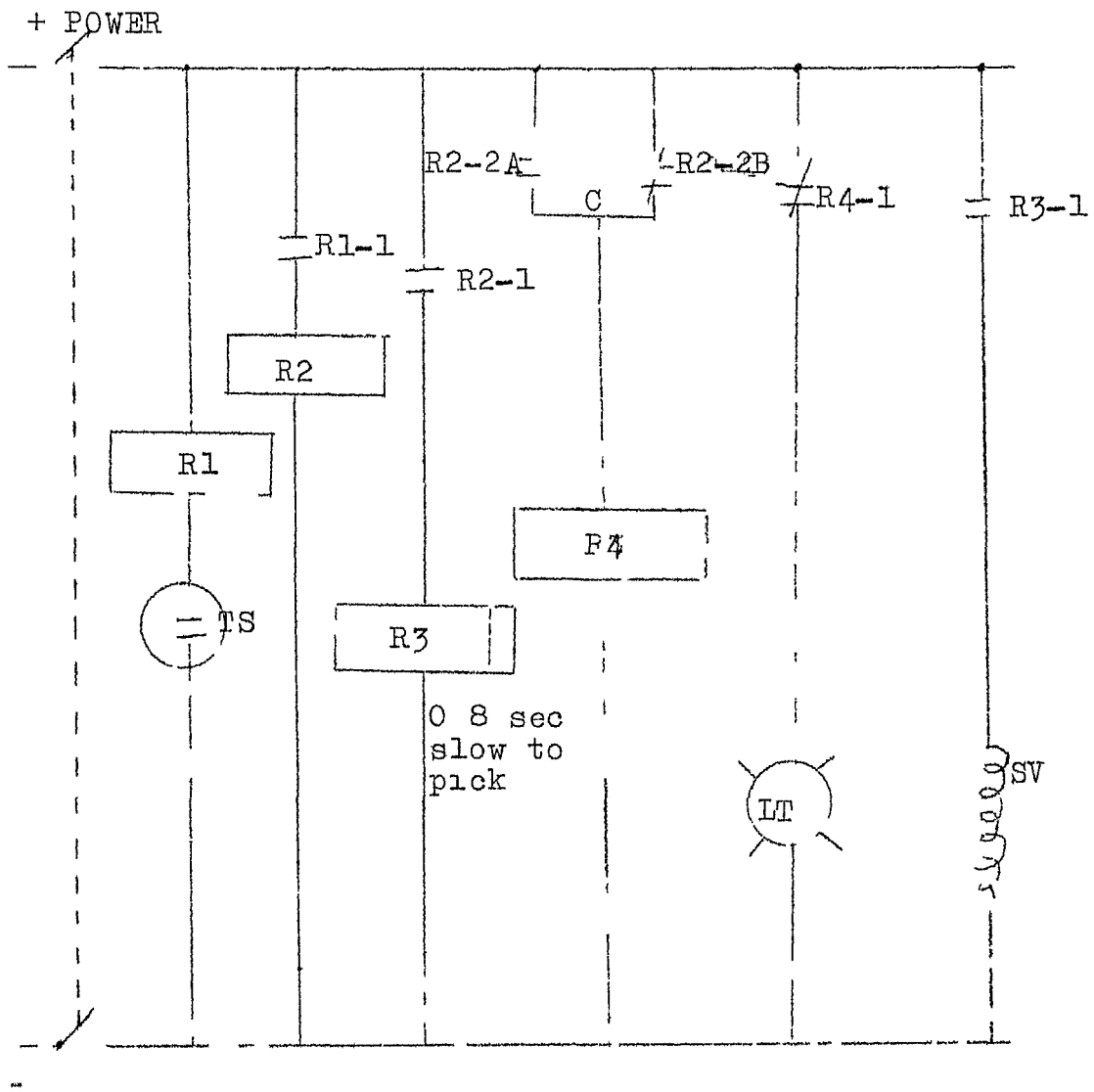

+ POWER
R2-2A
C
R2-2B
R4-1
R3-1
R1-1
R2-1
R2
R1
R4
TS
R3
0 8 sec
slow to
pick
LT
SV

Figure 7

CHAPTER 3

DATA STRUCTURE

3 1 INTRODUCTION

Choice of a data-structure is influenced by the considerations of storage (memory) and the kind of operations we want to perform on the data The data structure chosen should be such as to lend itself to the development of efficient algorithm for the operations to be performed In this chapter we will describe the data structure used for the relay circuit representation Followed by this we will describe the data cards

3 2 REPRESENTATION OF RELAY CIRCUIT

The operations that we have to perform on the circuit are -

(i) Update status of input contacts in accordance with input events

(ii) Given the status of all the contacts determine status of all the relay coils

(iii) On a change of the status of a relay coil update the status of its contacts

(iv) Given the status of all the contacts determine the status of all the output devices

From this it is evident that we must know, at all times, the status of every element in the circuit Referring to point (ii), we observe that we can determine whether a relay coil is energized or not if we

can determine whether a current is passing through it or not This problem can be reduced to the following form

'Given the status of all the contacts in the circuit find whether a path exists between a vertex V1 and another vertex V2' In our case, vertex V1 will be a coil or output-device terminal and V2 will be O1 (positive power supply line) or O2 (= negative power supply line) Obviously, this requires knowledge of the interconnection of the elements in the circuit

A linked list structure for the representation of the circuit has not been used since the chief advantage of such a structure, i e , ease in altering the structure (addition and deletion of elements) does not apply here The elements (i e , their 'type') and their interconnections donot change anytime during the processing Only the status of the elements changes Construction of the linked list structure itself would have been a long process Also, the storage required would have been larger

In the data-structure chosen, we consider the relay circuit as an undirected graph Each electrical node in the circuit is allotted a unique number called vertex number Each element in the circuit is considered as an edge of the graph and is allotted an edge number

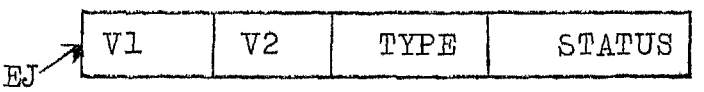
The above path finding algorithm can be solved if we know -

(a) Identify of all the edges connected to any vertex

(b) Identity of the two vertices across any edge

(c) Status of all the contacts, since a path can be extended through a contact only if it is closed

Each element in the circuit is stored as shown below



V1 and V2 are the two vertices across the edge No EJ TYPE is the type-code for the element, e g , an input contact, a relay coil, etc STATUS contains the current status of this edge

Of these three informations required, (b) and (c) are, thus, available in the element nodes

The information (a) is constructed internally by the program The edges connected to a vertex are called neighbours of the vertex The neighbours are allotted serial numbers in the order of their values Thus if edges 07, 12, 17 and 24 are connected to vertex V1, we consider 07 to be the first, 12 the second, 17 the third and 24 the fourth neighbour of V1 The information (a) is constructed in the form of a matrix NEBAR whose element NEBAR(V,N) is

the N^{th} neighbour of vertex V If we are at vertex V1, and e_1 is a neighbour of V1, with V2 as its other end, then we can reach V2 through e_1 or through any of the edges parallel to e_1 Thus if among a group of parallel edges, connected to a vertex V, we make only one of them, say e, to be the neighbour of V, and separately keep the information about the edges parallel to e, our purpose will be served This method saves a lot of storage Assuming a case when specifications permit, say 1000 elements, 99 vertices and we allow any vertex to have a maximum of 8 adjacent vertices and a maximum of 20 edges between any two vertices, we will require

$$99 \quad X \quad 8 X 20 \simeq \quad 16K \text{ words}$$

of storage for the matrix NEBAR In the alternative method, which has been adopted in this thesis, a vector PRLEJ contains the edge number parallel to the edges Thus the storage required is only 99 x 8 + 99 $\cong$ 1K words Also, in the alternative method there is no limit to the number of edges which may be parallel together. If edge numbers 03, 07, 11, 17 are parallel together and one end is connected to vertex V, then only edge 03, which is the lowest, is made neighbour of V And separately we have

PRLEJ(03) = 07
PRLEJ(07) = 11
PRLEJ(17) = 0

The value 0 for PRLEJ(17) signals the end of the parallel group If an edge e has no edge parallel to it, we would have PRLEJ(e) = 0

The above method can be employed for the storage of any undirected graph having a lot of parallel edges If n is the maximum number of vertices in the graph K is the maximum number of vertices adjacent to any vertex -e is the maximum number of edges in the graph Then the graph can be stored in

$$n \times K + 3e \text{ words}$$

For a graph having 100 vertices, 1000 edges, and K = 8, the storage requirement would be

$$100 \times 8 + 3 \times 1000$$

$$= 3800 \text{ words only}$$

It may be noted that there is no limit to the number of edges between any two vertices (i e , parallel edges)

The above method lends itself to an efficient path finding algorithm and requires less storage than those described in Reference 2 (Chapter 11, N Deo), if there also one word is used for storing each element of the matrix

3 3 0 Data Cards Now we will describe the data cards This information is essential for the understanding of the details of the algorithm described in next chapter.

The data cards contain information about the elements of the circuit, their interconnection, the input sequence and some other control and auxiliary information From this information, program determines the successive states of the circuit and prints out the same For the examples of the preparation of data cards, refer to User's Manual Fields on all data cards start from first column

3 3 1 Element Cards These are the data cards describing the relay circuit One card contains information for one element The fields on these elements cards are as follows

EJ, V1, V2, TP, RE, NAMENAME

EJ This is the two digit edge number allotted to the element Due to certain print-out limitations, this number is limited to from 01 to 90 Thus we can have a maximum of 90 elements in the circuit

V1,V2 These are the two digit vertex numbers across the element These can be from 03 to 99 No 01 and 02 are reserved for the two power supply lines Of the two vertices across an element any one can be considered V1 and the other V2, except in case of diode where V1 must be the vertex to which its anode is connected

TP This is a two digit integer code for the type of the element The various element types and their TP codes are as follows

Element	TP Code
Relay contacts	
Normally open	01
Normally closed	02
Form C - part A	03
Form C - part B	04
Form D - part A	05
Form D - part B	06
Any input contact	11
Relay coil	21
Output Device	31
Diode	41
Resistor	51

RE This is 00 for all elements except for relay contacts For relay contacts it is the two digit edge number of the relay (coil) to which the contact belongs This field is used to link the relays with their contacts NAMENAME This is an 8 character field containing name (label) of the element This is useful in identifying the element by its label

3 3 2 TDR Cards These cards are to be placed only for any TDR's in the circuit One card is required for each TDR Various fields on TDR cards are as follows

EJ, PICKD, DROPD

EJ is the two digit edge number of the TDR coil PICKD is the pick-up delay of the TDR, in what-ever units applicable (seconds, milliseconds, etc) is to be

given nearest to one decimal place, i e., it has a format ddd d (=F5 1)

DROPD is the drop-out delay of this TDR Its format is also F5 1

3 3 3 ADO Card Unless this is placed, status of only the output devices (TP = 31) will be reported during the analysis However in case status of some other elements, e g , some relays, is also required, these can be specified in ADO card Fields on this card are as follows EJ, EJ, EJ, , where each EJ is the edge number of the element whose status is also required

3 3 4 Input Events Cards These describe the initial status of all the input contacts and any subsequent changes These are prepared according to whether there are or not any TDR's in the circuit

When there are no TDR'S Each input event is coded on two cards The first card, called, Event Description card contains the description of the event, e g ,

THEN RESET PB IS PRESSED

The second card called EVENT DATA card contains the edge numbers and the new status of the input contacts affected by the event This is arranged as follows EJ = s, EJ = s, , where s is the new status of the edge number (±EJ) preceding s All the new status on an event-data card are considered to have occurred

simultaneously, and, therefore, the order of the edge numbers is not important

When there are TDR's In this case each input event is described by three cards Two of these are identical to those for the case of no TDR's The third card, called Time Card, is placed between the Event Description Card and the Event Data card It contains the lapse of time since last input event and is to be specified in the format (F5 1), ddd d

For the initial inputs, this is 000 0

Thus time cards contain the interval between successive input events

3 3 5 Option Cards These cards are placed to indicate whether or not

(1) List of elements is required

(2) Print out of topology of the circuit is required

(3) Diagnosis print-out is required

(4) Any TDR's are present in the circuit

(5) Any additional outputs are required

For details of these options and the cards see User's Manual

3 3 6 Initial Comment Cards These contain the initial comments on the job, as desired by the user These comments may include title of the circuit, name of the draftsman, date, etc Any number of cards can be placed. Contents of these cards will be printed as

such with a displacement of 10 positions to the right

3 3 7 * CARDS

These are the *1 to *8 and 99* cards and include option cards (Section 3 5) and some control cards All of these must be placed

See User's Manual for the sequence in which the data cards are to be placed

-

CHAPTER 4

IMPLEMENTATION DETAILS

4 0 INTRODUCTION

In this chapter we will describe in detail the program developed for the verification of the logical design of the relay circuits First an overview of the algorithm employed will be provided Following this will be the detailed algorithm At the end we will describe major subroutines and arrays used in the program

4 1 OVERVIEW OF THE ALGORITHM

From the information on element cards, the program constructs the internal representation of the circuit as two arrays Various tables are also prepared to facilitate further processing Then the circuit is put in no-power state Following this, if it is a circuit having no TDR's, the status of inputs is read, the input contacts' status updated, power switched on, and the resultant state of the circuit determined. The status of inputs and the resultant status of outputs is printed out This print-out is called Analysis Report Also included in the Analysis Report is the information as to whether the state reported is a transient state or a steady state After a steady state

is reported, next input is read and the processing done as above Processing is stopped when a *8END card is read If the circuit has TDR's then following the setting of circuit to no-power state, further processing is similar to the case of no TDR's except that now we require the precise moment of occurrence of the input status changes The Analysis Report contains information regarding the run-time (the lapse of time since power was switched on) and the interval between successive input events

4 2 THE ALGORITHM

Now we will describe in detail the algorithm employed for the verification of logical design of relay circuits For additional details of a step, reference should be made to the sections quoted in parentheses Steps 1 and 7 provide auxiliary information and donot constitute a part in the design verification We will illustrate various steps for the test circuit No 2 (Having TDR's), Figure 12. The flowchart for the Algorithm is shown on pages 40(a) to 40(f)

1 Read and print initial comments. (This results in first 3 lines of the output)

2 Read options.

3 Read Element cards

4 Prepare various arrays (tables) i e , INA, COILES VACS, VBCS, OPA, VAODS, VBODS and KTE (Section 4 4)

5 Construct internal representation of the circuit, i e , prepare arrays RRLEJ and NEBAR (Section 3 2)

6 If there are any TDR's, read TDR data cards

7 Print out, if demanded by an option, information about the elements and the topology of the circuit (User's Manual) This results in the LIST OF ELEMENTS and DATA ON TIME DELAY RELAYS, and DIAGRAM INFO Part A and B

8 Determine the connection form for each coil i e construct the vector EXCASE (Section 4 4)

9 Read the additional outputs, if any, and include these in the array OPA (In the test case additional outputs are the relays R1, R2 and R3)

10 Put the circuit in no-power state, i e , all the relays are de-energized and their contacts put in their normal states (This step employs subroutine KFS)

COMMENT Now the inputs will be set to their initial states and power switched on Analysis Report for the circuit having no TDR's is obtained by execution of Steps 12 1 to 12 8, whereas for the circuits having TDR's it is obtained by the execution of Steps 13 1 to 13 16 Steps 12 1 to 12 8 constitute subroutine ANLYS1 whereas Steps 13 1 to 13 16 constitute Subroutine ANLYS2 (Steps 12 1 to 12 8 can be considered as a subset of steps 13 1 to 13 16 As such we will illustrate only the steps 13 1 to 13 6 for the Test Circuit No 2)

11 If there are any TDR's in the circuit go to Step 13 else continue to Step 12 1

12 12 1 Constitute print formats for the Analysis Report Print the Report Headings

12 2 Read next input event description If it is end of events (i e , *8END card) print //END OF ANALYSIS// and stop Else continue

12 3 Read input event data card (Section 3.3 4.4)

12 4 Update status of input contacts in accordance with 12 3 (This step is executed by subroutine UPISTA)

12 5 Set KOUSC (CN)=0 for all the relays (KOUSC(CN) is the count of status changes of the coil number CN It is reset to zero at the beginning of the determination of next steady state, and is incremented, when there is a change of the state of the coil If the next steady state is not reached eventhough KOUSC(CN) becomes 5, it is assumed to indicate occurrence of cycling in the circuit Then a message is printed and processing terminated)

12 6 (This step is executed by subroutine SCANCL) Determine next state of the circuit (Section 4.5.2)

12 7 Report the state reached

12 8 If the state reported is a steady state go to Step 12 2 (to read next input event), else, if it is a transient state, go to Step 12 6

13 13 1 Construct print formats for the Analysis Report Print the Report headings (This results in the print out of 'ANALYSIS' and next 3 lines) Set RUNT = 0 0 (RUNT is run-time).

13 2 Read next input event description If it is end of input events (i.e , *8END card) print //END OF ANALYSIS// and stop Else continue to Step 13 3

13 3 Read and store in TLI the interval between the occurrence of this input event and its predecessor Set RTE = TLI (RTE is the remaining time for the occurrence of the input event For the initial inputs TLI=0 0)

13 4 Read input event data card

13 5 Update status of input contacts (First time, it makes EJST(1) = 0 on second input event EJST(1) = 1, etc)

13 6 Set KOUSC(CN)=0 for all the relays

13 7 Determine next state of the circuit

13 8 Report the state reached (This results in the line 2, for the first time)

13 9 If the state reported is a transient state go to Step 13 7 If it is a quasi-steady state go to Step 13 10 If it is a steady state go to Step 13 2.

13 10 (Now read the next input event and, if it is not the end of input events, determine whether this input event or the TDR with smallest waiting time *, called here SWR, should be attended to first, or should they both be attended to simultaneously)

Read next input event description If it is end of events go to Step 13.16 Else print input event description Read time and input Event Data cards Set RTE = TLI

13 11 Print information about the TDR's which are in timing state 8 or 9. (This information is printed in the form of identity of the TDR's, their states and

* At a given instant waiting time of a TDR is the interval after which it will time out and pick up or drop out, as the case be For example if a TDR, say R5, having a pick up delay of 1 5 seconds goes to state 8 at RUNT= 010 0 seconds, then at RUNT=010.5 waiting time of R5 will be 1.0 seconds

the waiting time This print-out helps in getting a better picture of the state of the circuit at the moment) (This results in the print-out of the lines 3,4,5)

13 12 (SWT is the smallest waiting timing i e , waiting time of SWR) -
If SWT > RTE go to Step 13 15 1
If SWT = RTE go to Step 13 14 1
Else (SWT < RTE) continue to Step 13 13 1
(For example at RUNT = 10 0, we have SWT= 0 5, and RTE = 0 3, so we go to 13 15 1)

13 13 1 (SWR is to be attended first)
Set RUNT = RUNT + SWT and RTE = RTE-SWT
Print RUNT and identity of the SWR which timed out at this moment (In test case, line 6 is printed) Update the status of SWR and its contacts and remove it from the queue of waiting TDR's (done by sub-routine SERADQ) Reduce waiting time of the TDR's by an amount equal to SWT

13 13 2 Set KOUSC(CN) = 0 for all the relays

13 13 3 Determine next state of the circuit

13 13 4 Report* the state reached

13 13 5 If the state reported is a transient state go to step 13 13 3

If it is a quasi-steady state, go to Step 13 11

(Else it is a steady state we should now attend to the input event) If the input event was 'end of input' (i e , *8END card) print //END OF ANALYSIS// and stop Else print input event description and go to Step 13 5

13 14 1 (SWT=RTE This means that the SWR times out precisely at the moment of occurrence of the input event). Set RUNT = RUNT + SWT

13 14 2 Print information about the identity of the SWR which timed-out at this moment (=RUNT) Also print description of the simultaneous input event

13.14 3 Update status of SWR and its contacts Remove SWR from the queue of waiting TDR's Reduce waiting time of all remaining TDR's in the queue by an amount SWT Update status of input contacts in accordance with input events. Go to Step 13 6

13 15 1 (SWT > RTE Hence input event occurs before SWR times-out) Print input event description

13 15 2 Update status of input contacts

13 15 3 Set RUNT = RUNT + RTE

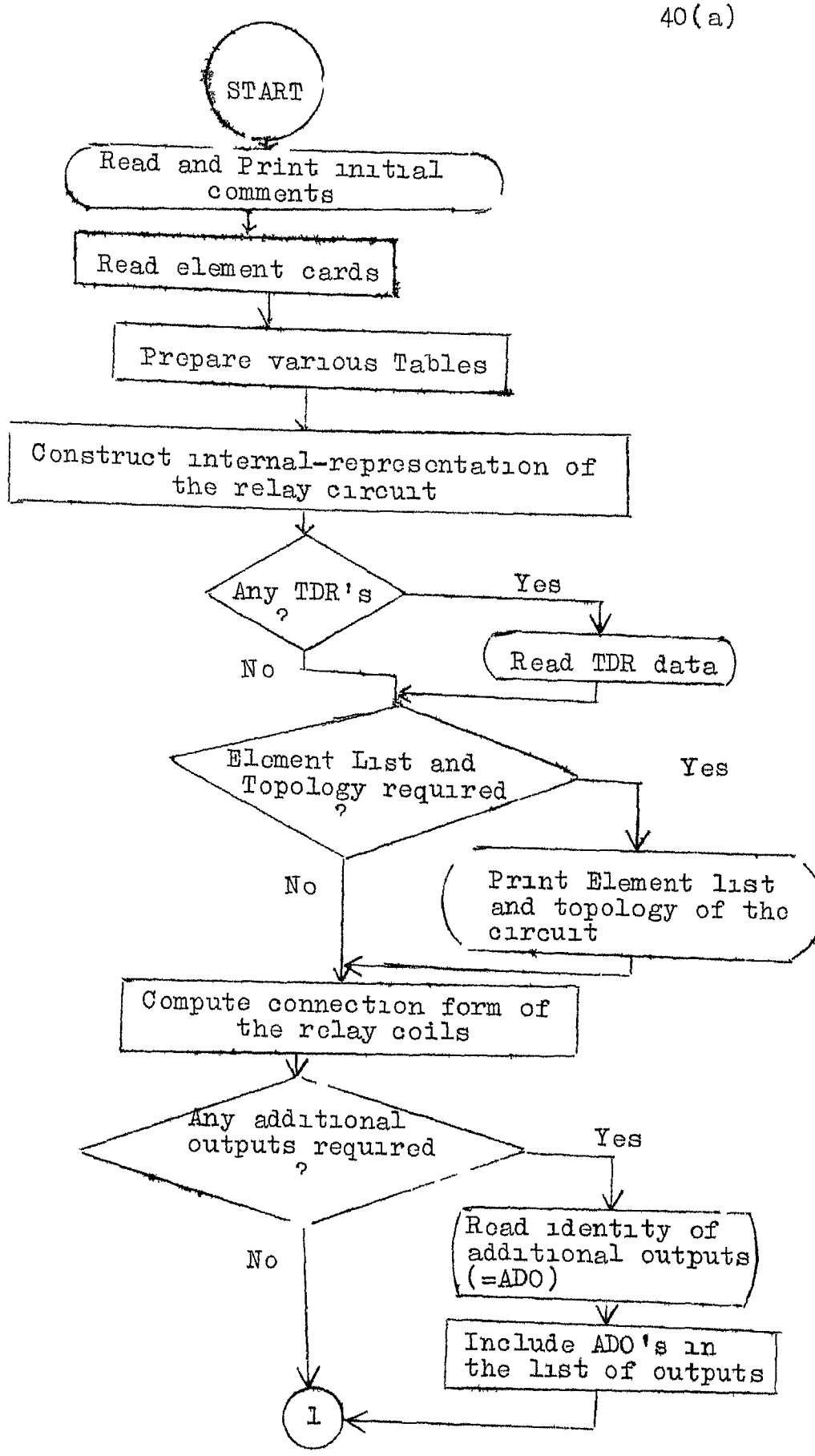
13 15 4 Reduce waiting-time of all the waiting TDR's by an amount equal to RTE Go to 13 6

13 16 (Input events have ended-up, but some TDR's are in timing-states Effect of timing out of these TDR's should be studied) Set RTE = 99999 9
Go to 13.11

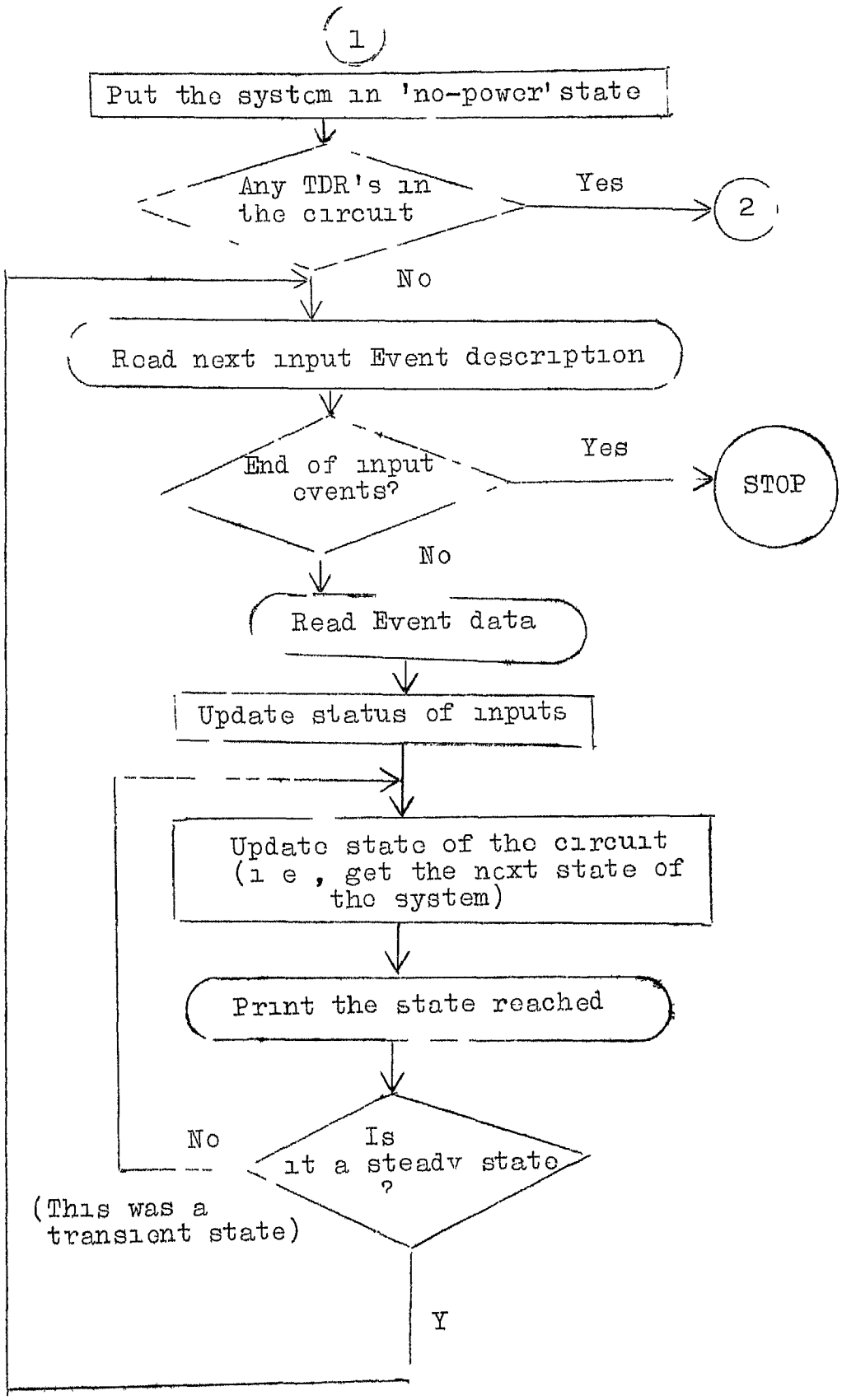
End of Algorithm

4 3 DIAGNOSIS

If demanded by the user by an option card (*4D card), status of all the relay coils and their contacts is printed out, after the print out of each state This print out is called Diagnosis print out Diagnosis print out in conjuction with the print out of the state, gives us

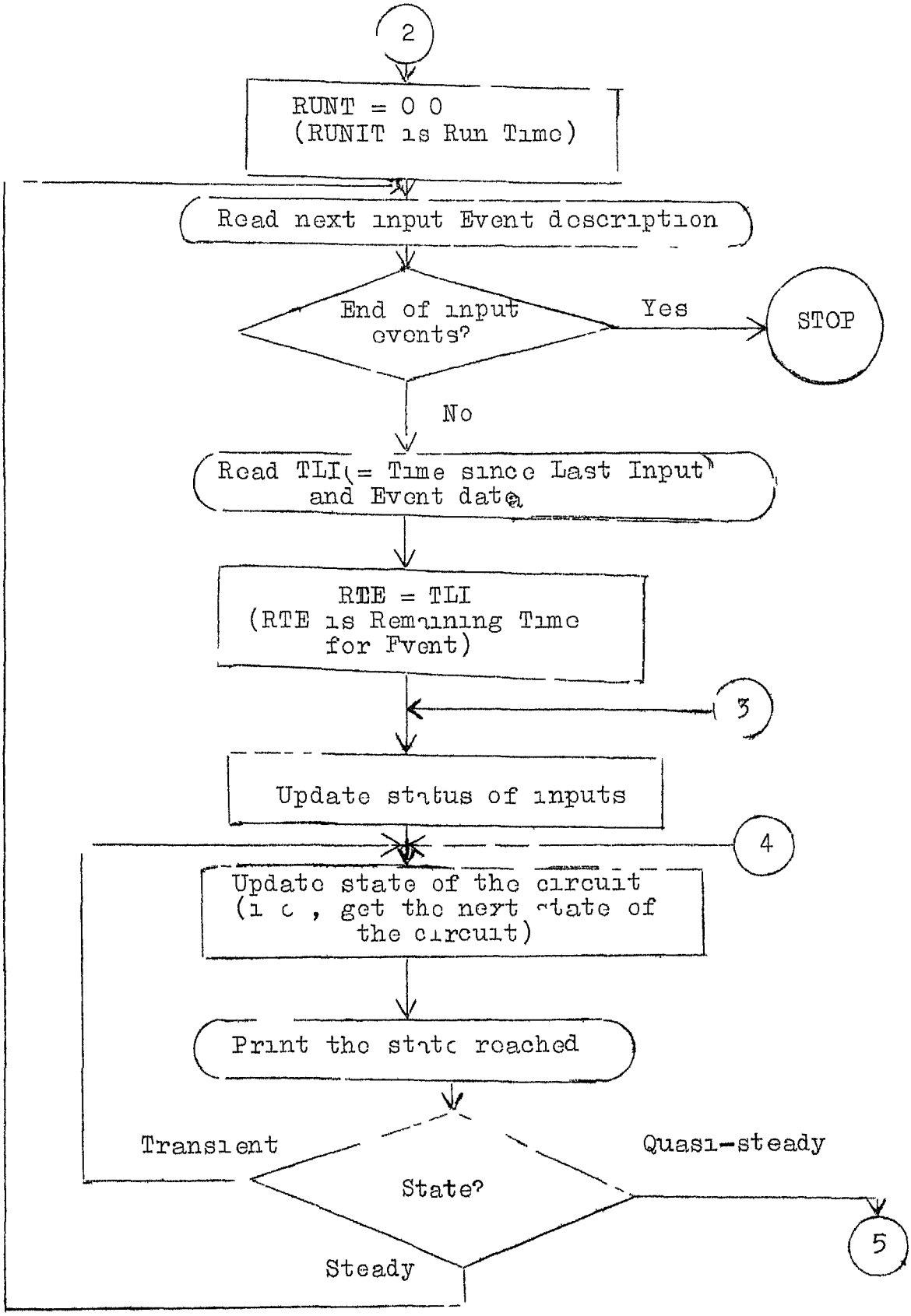


Analysis Algorithm - Flow Chart -continued

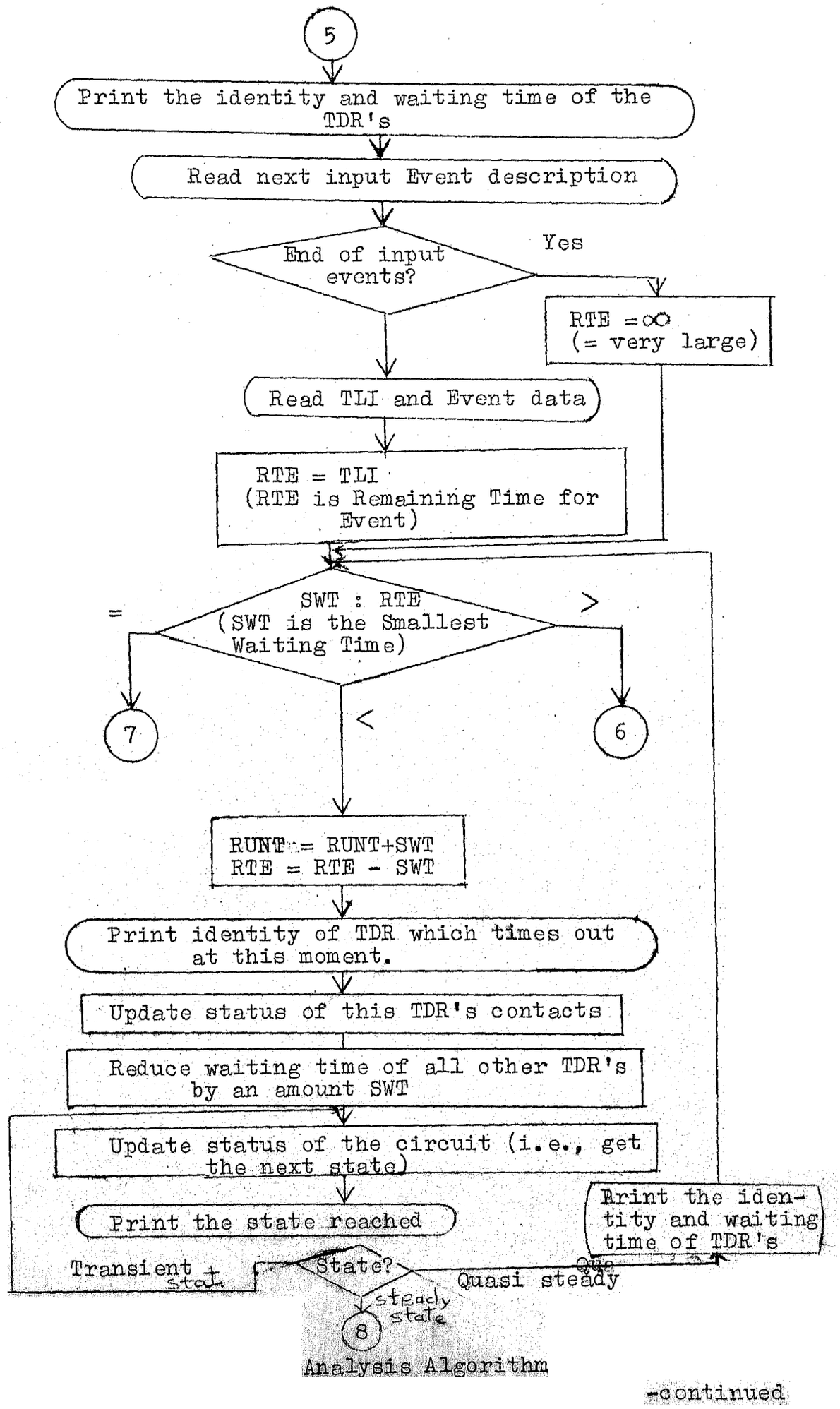


Analysis Algorithm

-continued

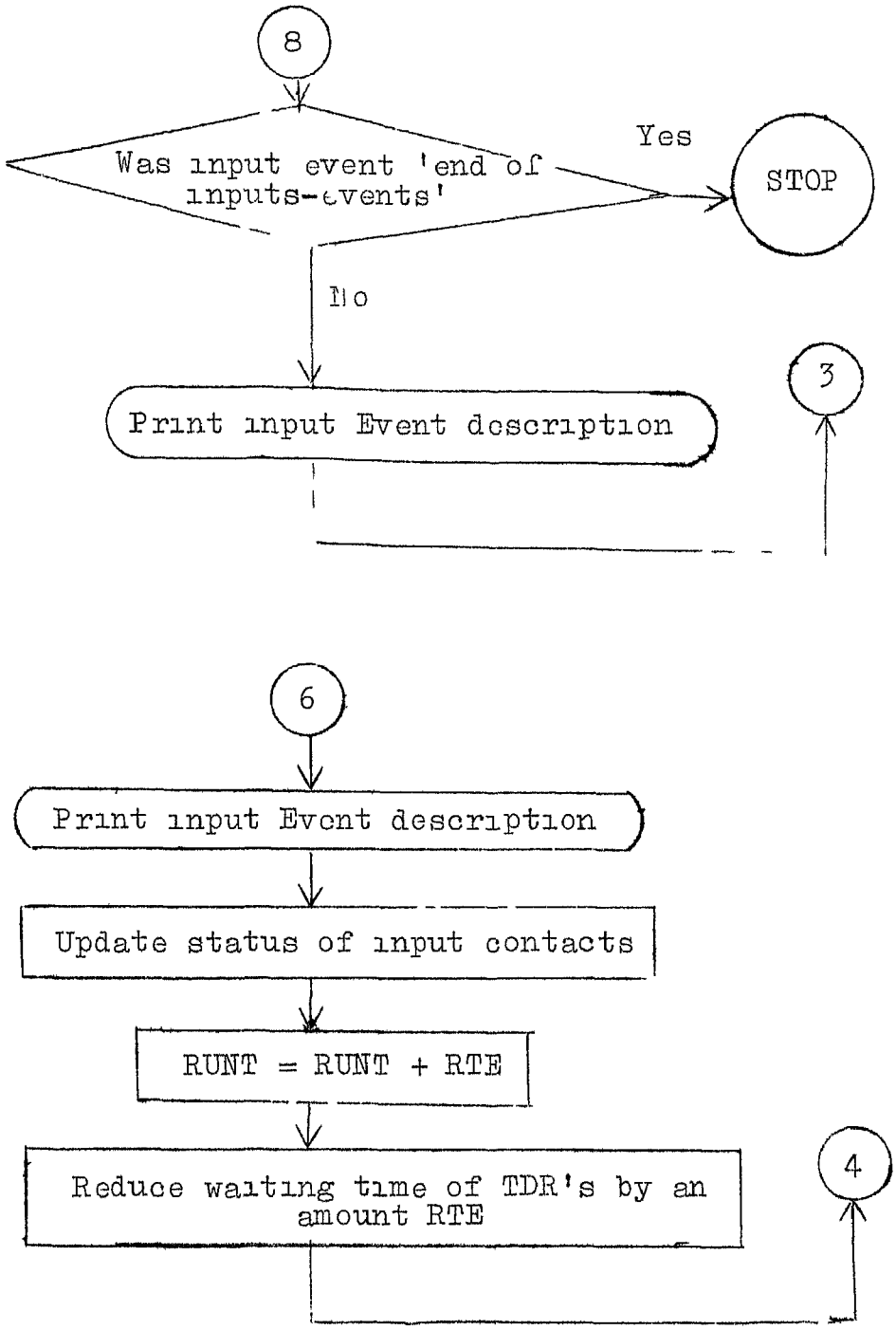


Analysis Algorithm



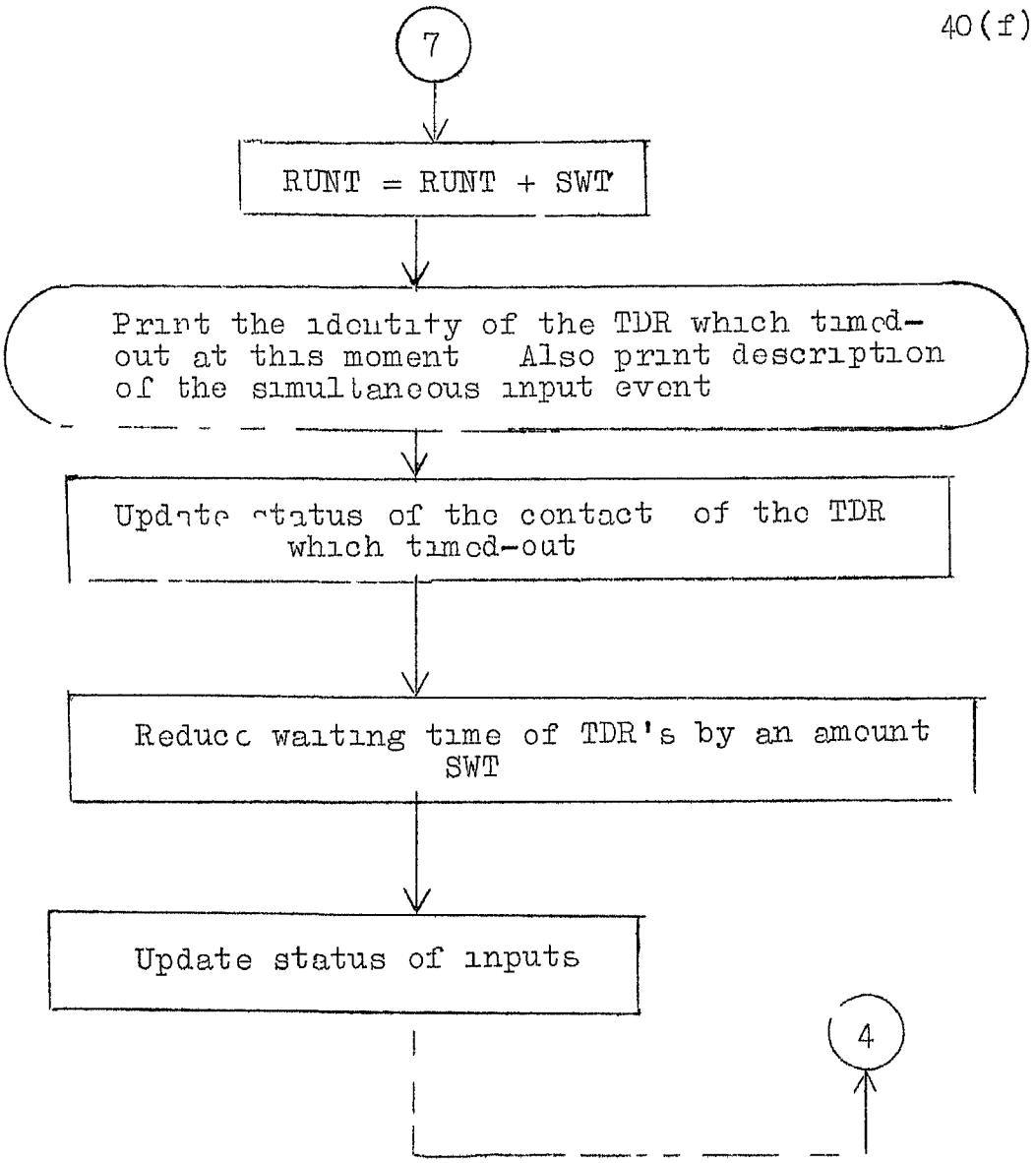
Analysis Algorithm

-continued



Analysis Algorithm

-continued



Analysis Algorithm

status of all the elements in the circuit Knowing the status of all the elements in the circuit, at successive stages of the Analysis, we can determine the cause of the error in the design of the relay logic circuit

4 4 DESCRIPTION OF ARRAYS USED IN THE PROGRAM

4 4 1 INA This is a vector and contains edge numbers of the input contacts INA has a dimension of 16, which is the maximum number of input contacts allowed in the circuit INA is useful in updating status of the input contacts It is also used in printing headings of the Analysis Report (For the example circuit, Figure 12 , we would have, INA(1) = 2, INA(2) = 0, INA(3) = 0, , INA(16) = 0)

4 4 2 OPA This is similar to INA and contains edge values of the outputs (In our case, OPA(1) = 8, OPA(2) = 10, OPA(3) = 0, OPA(4) = 0, , OPA(16) = 0)

4 4 3 COILS This vector contains the edge numbers of the relay coils Its dimension is 18 corresponding to a maximum of 18 coils allowed in the circuit This is used at many places in the program (For the test case, COILS(1)= 1, COILS(2)= 3, COILS(3)= 8, COILS(4)= 0, , COILS(18)= 0)

4.4 4 VACS This vector contains the vertex number to which terminals A of the coils are connected Thus VACS(1) is the vertex number to which terminal A of the coil number 1 is connected This is useful in

determining status of the relay coils (For the test case VACS(1) = 1, VACS(2) = 1, VACS(3) = 6, VACS(4) = 0, , VACS(18) = 0)

4 4 5 VBCS This is similar to VACS and contains vertices to which terminals B of the coils are connected (For the test case VBCS(1)=3,VACS(2) = 4, VACS(3) = 2, VBCS(4) = 0, , VBCS(18) = 0)

4 4 6 VAODS and VBOBS These vectors contain information about the output devices and are similar to VACS and VBCS respectively

4 4 7 KTE This is a two dimensional array It contains the edge numbers of the contacts of the relay coils KTE(CN, N) is the edge number of the N^{th} contact of the coil number CN It has the dimensions of (18,12) correspond to a maximum of 18 coils and a maximum of 12 contacts allowed in a relay This array is useful in updating the status of the contacts of a relay when there is a change in the state of the relay (For the test case, we would have

KTE(1,1) = 4, KTE(1,2) = 7, KTE(1,K) = 0 for K=3 to 12
KTE(2,1) = 5, KTF(2,K) = 0, for K = 2 to 12
KTE(3,1) = 9, KTE(3,K) = 0, for K = 2 to 12)

The arrays 4 4 1 to 4 4 7 are prepared by scanning all the edges once Array KTE is prepared by scanning the edges once again

4.4.8 EXCASE To describe this array, first we will explain the 'connection forms' of the relay coils See Figure 8(a) We can determine whether a relay coil is energized or deenergized by finding the state (i.e., conducting or not) of the four extensions possible from the coil terminals A and B to the power supply lines These extensions are (A to 01), (B to 02), (A to 02) and (B to 01) Calling these extensions as K1, K2, K3 and K4 respectively, we see that the state (=E) of the coil is given by the logical equation

$$E = (K1 \wedge K2 \wedge \overline{K3} \wedge \overline{K4}) \vee (\overline{K1} \wedge \overline{K2} \wedge K3 \wedge K4)$$

KX = 1 implies KX is conducting

E = 1 implies the coil is energized

The state of the extensions K1, K2, K3 and K4 can be found by subroutine PATH

In the above method we have to determine all the four K's for each of the coils Since status of each coil is determined several times during the Analysis, and also since the path finding algorithm used to determine the state of these K's is itself a time consuming process, we should see if we can ignore some of the K's, wherever possible, for some of the coils It is found that this indeed can be done if we know the connection form of the coils

The four connection forms of the coils are shown in Figure 8 The Table No 2 gives the extensions relevant to each connection form and the relation between the state of the coil and its extensions

Table No 2

Connection Form	Extensions relevant	E
1	K1,K2	$E = K1 \wedge K2$
2	K1, K2,K3	$E = K1 \wedge K2 \wedge \overline{K3}$
3	K1,K2,K4	$E = K1 \wedge K2 \wedge \overline{K4}$
4	K1,K2,K3,K4	$E = (K1 \wedge K2 \wedge \overline{K3} \wedge K1) \vee (\overline{K1} \wedge \overline{K2} \wedge K3 \wedge K4)$

Determination of Connection Form To determine the connection form of the coils, all the contacts in the circuit are set to 1 Then, for each coil the existence or not of K3 and K4 is found by determining if there exists a path (transmission) from terminal A to vertex 02, and from terminal B to vertex 01 respectively By knowing whether K3 and/or K4 exists for the coil, we know its connection form

Value of the connection form (=1,2,3 or 4) of a coil CN is stored in EXCASE (CN) (For the test case, connection form of all the coils is 1)

While determining the required state of the coils (see subroutine SCANCL, Section 4 5 2) each coil is processed in accordance with its connection form Most of the relay coils in process control switching circuits have the connection form 1, i.e , only two of the four possible

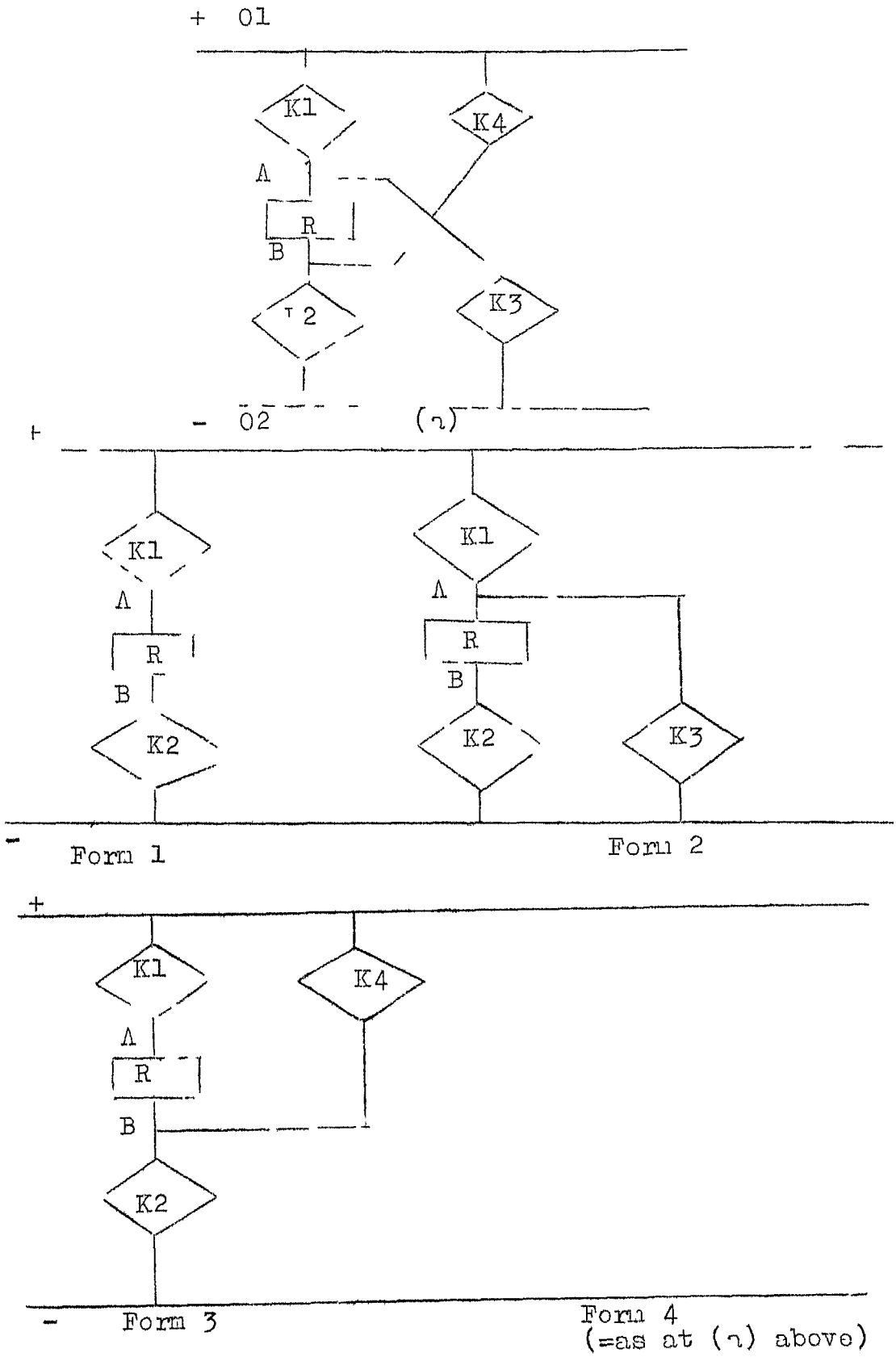


Figure 8 Connection Forms of Relay Coils

extensions are relevant for them Thus by knowing the connection form of the relay coils, the execution time for determining the states of all the coils is reduced to, approximately half as compared to the method where we would consider all the four extensions possible for each coil

4 5 SUBROUTINES

In this section we will describe the major subroutines used in the program There are a total of 13 subroutines

4 5 1 Subroutine PATH This is a very important subroutine and is called several times during the preparation of arrays and the Analysis

Given the status (0/1) of all the contacts in the circuit, it determines whether or not there exists a transmission between two specified vertices SV and GV The algorithm used is as follows (Variable LUCK is set to 1 if transmission exists between SV and GV, else it is set to 0)

P1 If SV = GV, then set LUCK = 1 and return

P2 (Unmark all the vertices)

Set MARK(V) = 0 ∀ V

P3 Set VX = SV

(VX is the vertex reached)

P4 Set STP = 0

(STP is the stack-pointer of a stack VSTK, which holds the vertices through which the path has been extended to the current vertex VX The vertices in the stack are stacked in the order in which they were traversed, starting with SV at position 1 and VX at the top position STP)

P5 (Push VX on the stack)

STP = STP + 1

VSTK(STP) = VX

MARK(VX) = 1

NEBN(STP) = 1

(NEBN(STP) is the neighbour number of the vertex (refer Section 3.2) at position STP in the stack)

Set NB = NEBN (STP)

P6 Set EJT = NEBAR(VX,NB)

(EJT is the edge number of the NBth neighbour of the current vertex VX EJT = 0 implies we have already attempted all the neighbouring edges of VX to extend the path from VX to a new vertex, but without success (i e , all the neighbouring edges are open) Hence we should try to extend the path through the next neighbour of the earlier vertex)

IF EJT $\neq$ 0 Go to P8

Else (if EJT = 0), set STP = STP - 1

IF STP $\neq$ 0 Go to P7

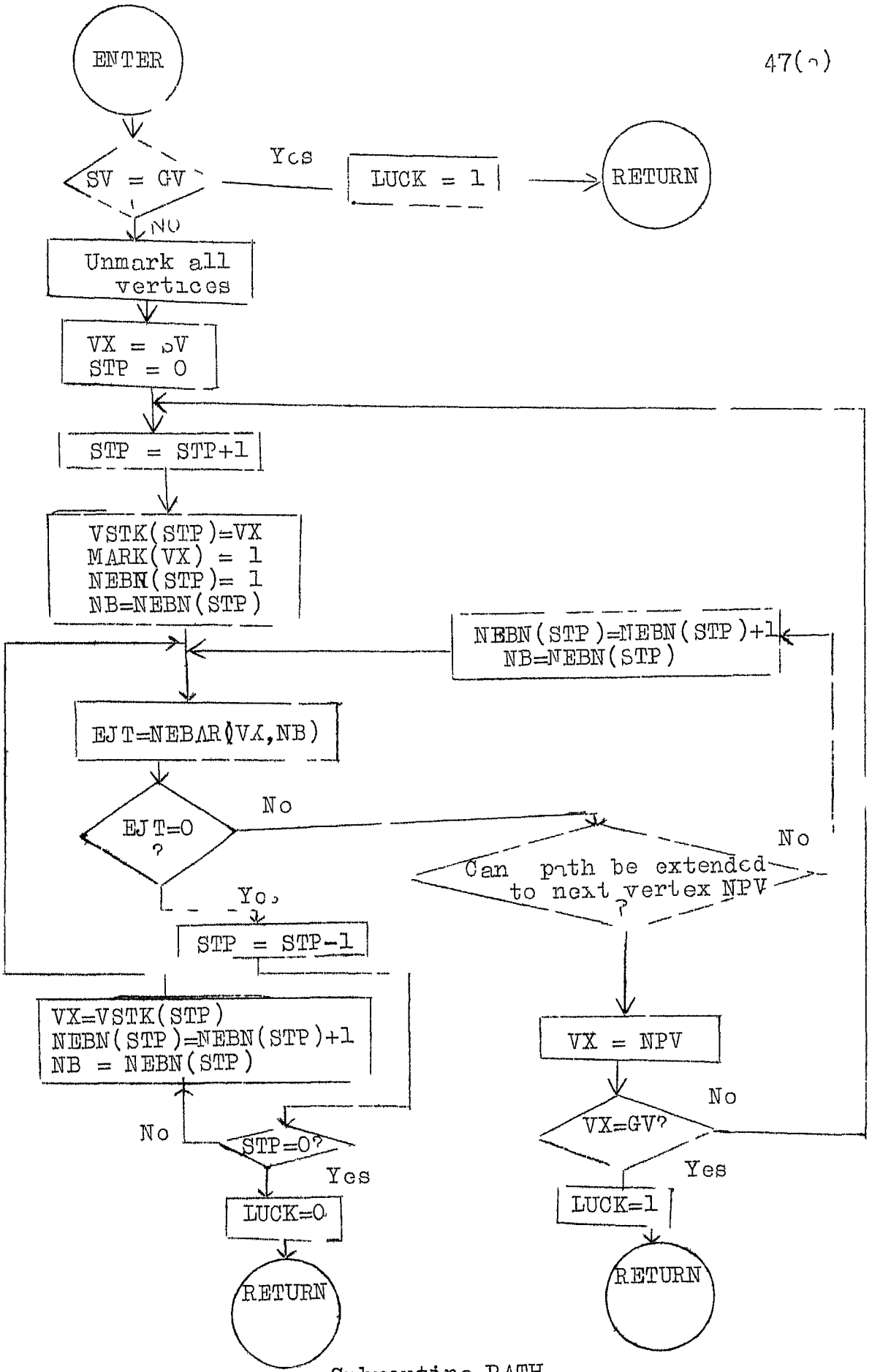
(Else stack has become empty on back-track This implies no path could be found between SV and GV)
Set LUCK = 0 and return

P7 (Back-track Try next neighbour of the earlier vertex)
VX = VSTK(STP)
NEBN(STP) = NEBN(STP) + 1
NB = NEBN(STP)
Go to P6

P8 (We will attempt to extend the path through EJT or edges parallel to EJT NPV is the vertex to which EJT, or its parallels, will lead us to from VX It may be noted that

(1) It is no use extending path to NPV if it is marked i e , has already been traversed earlier)
(2) We cannot extend path through a coil or an output device
(3) If an edge is a diode we can extend path through it only if the VX is anode when GV is C2, or if VX is cathode when GV = 01
(4) We can always extend path through a resistor
(5) Path cannot be extended through a contact which is open (i e , State = 0))

If path can be extended to NPV go to P9
(else try next neighour of VX)
Set NEBN(STP) = NEBN(STP) + 1
NB = NEBN(STP)
Go to P6



Subroutine PATH

P9 VX = NPV

If VX = GV set LUCK = 1 and return else go to P5

End of Subroutine PATH

4 5 2 Subroutine SCANCL This subroutine updates the status of relay coils and their contacts and the outputs, i e , it updates the circuit to its next state It also determines whether the state so reached is a transient state, a quasi-steady or a steady state It does all this in 4 parts

(1) First it determines the 'required' or the permitted state for each relay If the relay is de-energized, its required state is 0, and if it is energized, its required state is 1

(2) After determining the required-state of all the coils, it compares these with their present status and takes action as per Table No 3 By the end of execution of this part, the relay contacts have been set to their new states

(3) Then it updates the status of the outputs

(4) Then it determines whether the state reached is a transient state, a quasi-steady or a steady state

Now we will explain the above four parts

Required state of the coils are determined by determining the states of the extensions relevant to the coils (See Section 4 4 8)

Action on Relays

Initially set KTCH = 0 Later during the action on relays if the status of any relay's contacts is changed set KTCH = 1 The actions are as per Table 3

Table No 3

Previous status of coil CN =PRST(CN)	Required status of Coil CN =REQST(CN)	Action
0	0	Nil
0	1	(Relay has been energized from a dropped out state) If this coil has a pick up delay go to 1) Otherwise set KTCH = 1 Update its contacts+ status, setting transfer contacts, if there are any, to their transitory state Set PRST(CN) = 1 if this relay has no transfer contacts else set PRST(CN) = 2 (1) Enter this relay at the appropriate place in the queue of waiting relays (done by Subroutine ENTPRQ) Set PRST(CN) = 8
1	0	(Relay has been de-energized from a picked up state) If this relay has a drop-out delay go to(2) Else set ITCH = 1 Update its contacts' status, Setting transfer contacts, if there are any to their transitory state Set PRST(CN) = 0 if this relay has no transfer contacts else set PRST(CN) = 3 (2) As (1) above, but PRST(CN)=9
1	1	Nil

PRST(CN)	REQST(CN)	Action
2	0	(This relay has been de-energized immediately after it was energized, and its transfer contacts were still in transition Set all its contacts to their normal state Set KTCH = 1 and PRST(CN) = 0
2	1	Put its transfer contacts to their final state corresponding to State 1 of the relay Set KTCH = 1 and PRST(CN) = 1
3	0	Put its transfer contacts to their final state corresponding to State 0 of the relay Set KTCH = 1 and PRST(CN) = 0
3	1	(This is analogous to the case PRST(CN) = 2, REQST(0)) Set all its contacts to their picked up state Set KTCH = 1 and PRST (CN) = 1
8	0	(This relay, which was timing to pick-up, has been de-energized) Remove this relay from the queue of waiting TDR's (done by subroutine WTHDR) Set PRST(CN) = 0
8	1	Nil
9	0	Nil
9	1	(This relay which was timing to drop out has been energized) Remove this relay from the queue of waiting TDR's Set PRST(CN) = 1

Updating the Status of Outputs This is done by subroutine STTOPA, which is very simple and hence will not be described

Determination of the state of the circuit At the end of the action on all the relays, if KTCH = 0, then this implies that no changes in any relay contacts' status were made, and hence the circuit is not in a transient state now If there are any TDR's in the waiting queue, then it is a quasi-steady state, else it is a steady state

However, if KTCH=1, then this implies that some relay contacts were updated while taking action on relays We must see the effect of such actions on the state of relay coils Therefore this state is a transient state

46 Memory Requirements The program (listing appended) requires approximately 14K words of memory space in 7044 Data space for the specifications listed in Appendix A is approximately 3K Approximate requirements of data space can be computed readily from the DIMENSION statements

CHAPTER 5

CONCLUSIONS

5 1 RESULTS AND SUMMARY OF THE WORK

Objectives of the work were stated in Chapter 1 In accordance with these objectives a FORTRAN program has been developed and tested satisfactorily on IBM 7044 for many relay circuits As an illustration the results for two test circuits - Test circuit No 1 (Having no TDR's) and Test Circuit No 2 (Having TDR's) are appended Circuits having following features were also tested

(1) Having resistors

(2) Having diodes

(3) Having connection forms 2,3 and 4 for the relay coils

(4) Cycling

In some cases, errors were incorporated deliberately in the coding of the relay diagram but these were detected by the Element List, and structural print-out features One of the large systems tested was an Audio-Visual Annunciator The diagnosis print-out has been found to be of immense use in determining the cause of error in the design of the relay circuit The coding procedure is very simple making the analysis of the relay circuits a simple job.

In order to meet the fourth objective, namely, 'it (the program) should provide easily digestable printout' the Analysis Report has been arranged in the form of a table showing status of inputs vs resultant status of outputs This has resulted (due to limited width of the printer paper), in allowing only a maximum of 16 inputs and 16 outputs However, by adopting some other format for the Analysis Report, e g , status of inputs in one line followed by the status of the outputs on next line, we can easily allow a large number of inputs and outputs Upper limit on the number of other elements, as well as the maximum number of elements in the circuit, can also be increased thereby enabling one to analyse very large relay systems

Implementation of the specifications listed in Appendix A requires approximately 14K words of storage in 7044 For a relay circuit having 50 elements, 5 relays, 4 inputs and 4 outputs, and of average topological complexity, the analysis involving 10 input events would require approximately 10 seconds on IBM 7044

Verification of the coding of the circuit could have been done by printing out the circuit figure But this would have required, as input, the positional information of the elements which in turn would require standard drafting procedure for drawing the relay diagram with the elements on a grid structure (Although from the interconnection information alone, we can generate the figure, but it

would at times be very difficult to establish isomorphism between the printed figure and the input figure) In view of the above, the provision of 'element list' and the interconnection information in the form of a table is satisfactory

5 2 SCOPE FOR FUTUPE WORK

In the present work only those elements have been included which are mo t commonly found in process control relay switching circuits However, in certain applications there may be other types of elements in the circuit One such special element is the stepping switch A system can be developed whereby one can specify operation of any special elements and then he should be able to include it in the relay circuit Another useful feature would be to specify only the new position of a multi-position manual switch (when it is operated) rather than the status of each of its contacts.

For a wider application, e g , telephone switching circuits, one should include other types of relays, rather than only the 2-terminal type in the present work

-

APPENDIX A

Logical design of any relay circuit meeting the following specifications can be verified by the program However, the program can be easily altered for any changes in the specifications

SP1 The circuit must have at least one input-contact, one output device, one relay and one contact of each relay (If it is not meeting these minimum requirements, add the missing item with its both ends unconnected)

SP2 Any combination of relay coils and/or output devices in series is not allowed They can be in parallel

SP3 Output devices can be connected only in the forms shown in Figure 9 In this figure K's represent contact and diode logic or just a connecting wire Only reverse biased diodes can be connected across the output devices, as shown

SP4 Number of elements in the circuit ≤ 90 *

SP5 Number of relays in the circuit ≤ 18 *

SP6 Number of input contacts ≤ 16 *

SP7 Sum of the number of output devices and the additional outputs required ≤ 16 *

SP8 A relay in the circuit cannot have more than 12 of its contacts in the circuit (Note that each Form C or Form D contact is equivalent to 2 contacts)

---

* Lower limits are dictated by SP1

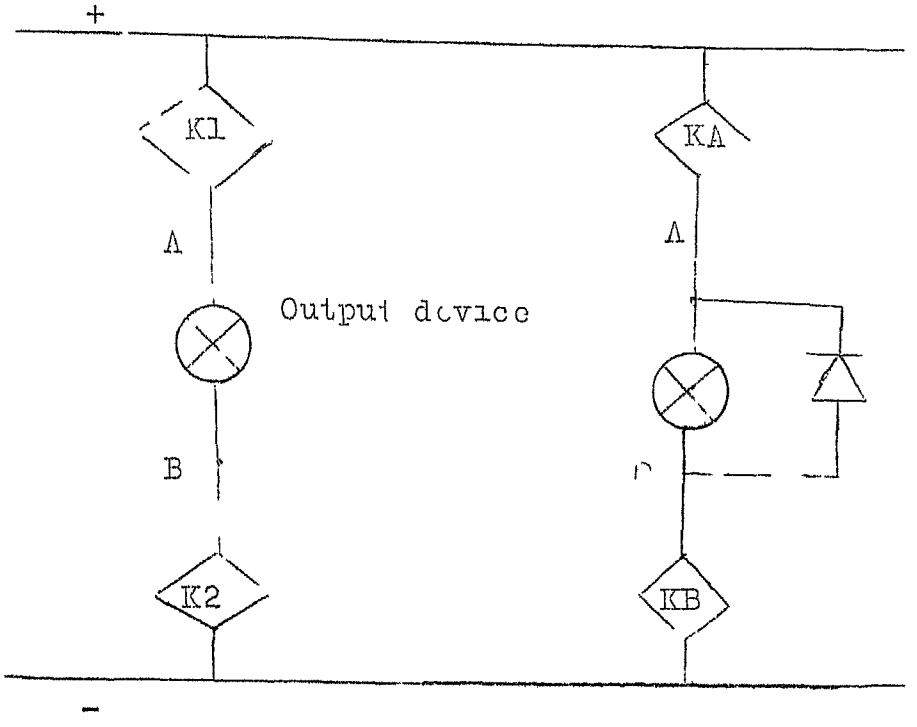


Figure 9

SP9 Except for the power supply lines (i e , vertex 01 and 02) no other vertex can have more than 8 adjacent vertices Program has a provision to detect violation of this specification and terminate processing with a message

SP10 Between any pair of vertices there can be any number of (i e , parallel) elements

SP11 Minimum pick up/delay of any relay = 000 0 time units
Maximum pick up delay of any relay=999 9 time units
These limits apply to drop out delay also

APPENDIX B

USER'S MANUAL

For verifying the logical design of a relay switching circuit by the program, proceed as follows

U1 Ensure that the specifications in Appendix A are not violated by the circuit

U2 Prepare data cards as described in Chapter 3 The data deck should appear as shown on Page 59 After you have prepared data cards for one or two circuits, there will be no need to refer Chapter 3

The points to remember while preparing the data cards are listed below

(1) Vertex No 01 and 02 must be allotted to positive and negative power supply lines respectively

(2) For diodes the anode vertex must be punched first on the element card, i e , should be V1

(3) A relay contact whose coil is not shown in the circuit should be considered an input contact and allotted TP code 11

(4) When a manually operated switch is actuated, the change of state of all the contacts affected should be specified in the input event data card For example in the circuit in Figure 10, when the switch HS is thrown to position OFF from ON, this event would be described as follows

HS PUT TO OFF

02 = 0, 04 = 0.

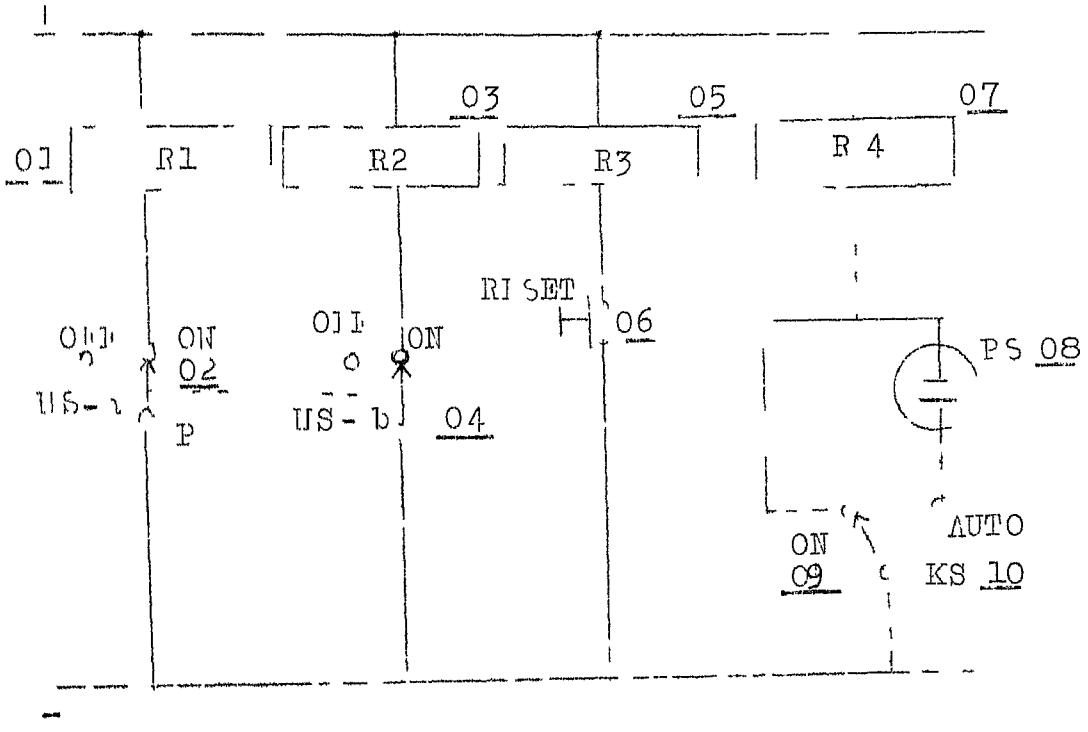


Figure 10 (This is only a part of a circuit)

(5) Actuation of a spring return push buttons will be described by two successive events - one corresponding to the pressed condition of the push-button and next the released condition For example when the RESET push-button is actuated in Figure 10 this will be described as

RESET PB PRESSED
06 = 1
THEN RELEASED
06 = 0

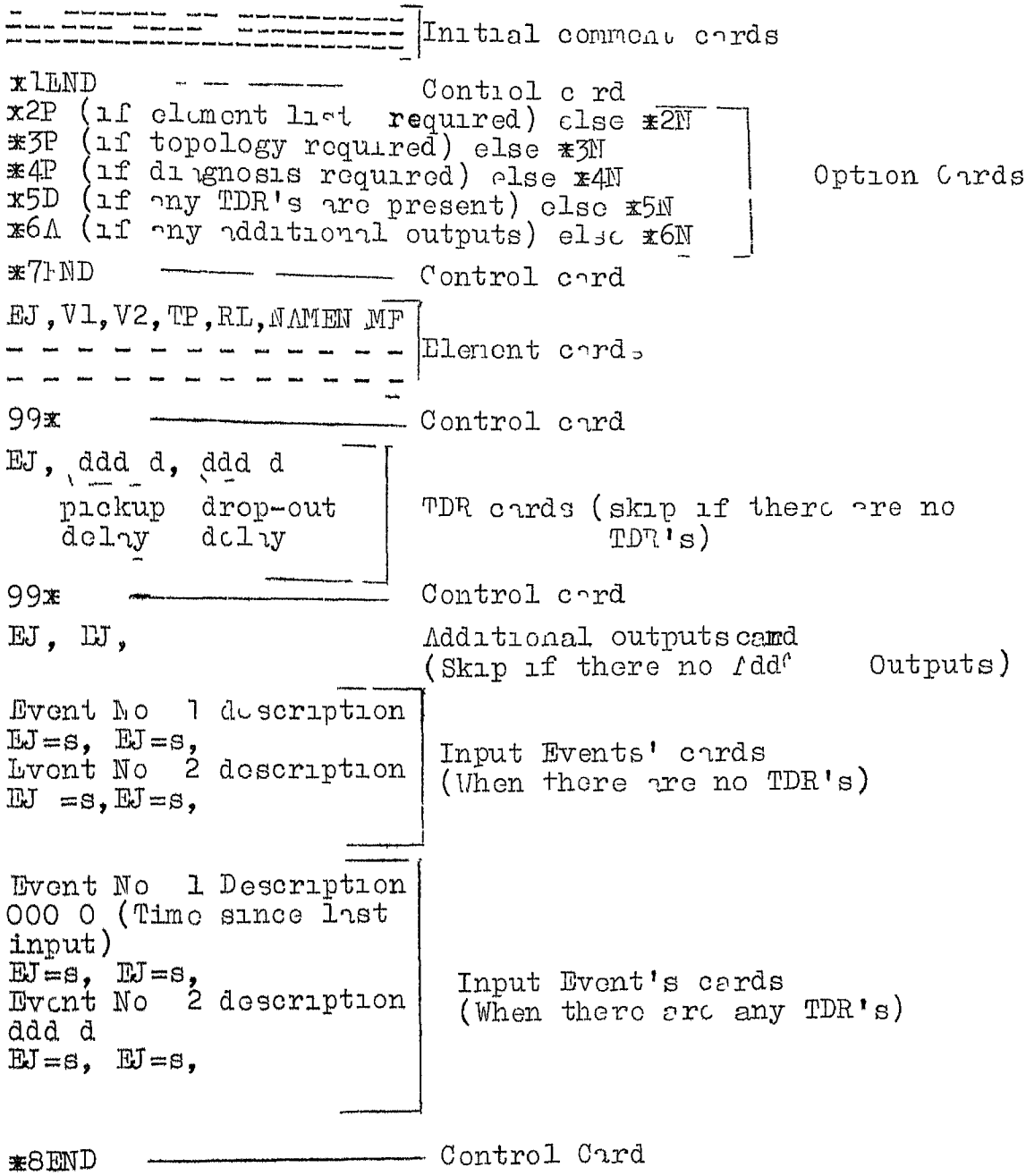
(6) In case of switches having transfer contacts the transistory state should be also described For example in Figure 10 when switch KS is thrown from position ON to AUTO, first the edge 09 opens then the edge 10 closes This will be described as follows

KS PUT TO AUTO
09 = 0
THEN
10 = 1

If the above were a continuity transfer contact (Form D) the event will be described as

KS PUT TO AUTO
10 = 1
THEN
09 = 0

Data cards for the Test Circuit No 1 (Figure 11) and Test Circuit No 2 (Figure 12) are shown on Pages 64 and 65 respectively



Arrangement of Data Cards

Interpretation of Print-outs Interpretation of various print outs is as follows For an example, refer to the results of Test Circuit No 1 (Figure 11) and Test Circuit No 2 (Figure 12), attached with the figures

(1) First the program prints-out a true copy of the Initial Comments cards,

(2) Then, if a * 2P card has been placed, it prints the list of elements in the same format as the Element Cards Followed by the list of elements is the information on TDR's (if there are any TDR's) If a *2NO card instead of *2P was placed, these two print-outs are not produced

(3) Then, if a *3P card was placed, it prints the information about the interconnection of the circuit elements This is in two parts PART-A is the list of edge numbers vs ed e numbers which are parallel to them When a number of edges are parallel together then an edge is considered parallel to the next lower edge number in the group Thus if edge Nos 01, 03, 11 and 15 are parallel together, this information will be printed as

```
EDGE     01  03  11  15
PARALLEL 03  11  15   0
```

PART-B is the table of vertex numbers vs the lowest edge numbers leading to the adjacent vertices

When a new circuit is taken up for analysis, the above print-outs should be demanded and correctness of coding and punching should be confirmed Once the element cards have been thus verified to be correct, the above print outs need not be demanded

After the above print outs the Analysis Report is produced We will explain the Analysis Report for the case of Test Circuit No 2 (Having TDR's) The first line in the report is

IN OUT

In means Input and OUT means output

In the next line

LN means input Event Number

RUNTIME is the lapse of time since the first input event.

TLIN is Time since Last Input i e , it is the interval between successive input-events Next to TLIN are the edge numbers of the input contacts STATE refers to the state of the circuit Next to STATE are the edge numbers of the outputs

After the print out of these headings, the input out/put report starts which is self explanatory However, following points will be of use to those interpreting results for the first time

(1) Description of input-events are printed as a true copy of the event-description card But it is preceded by the serial number of the input event, e g ,

1) INITIAL INPUTS

- - - - - - -
- - - - - - -

2) THEN TS CLOSED

- - - - - - -
- - - - - - -

(2) Whenever an input event occurs, the new status of all the inputs is printed vertically below the input edge numbers

(3) TRANS means a transient state (which lasts for only about 10 milliseconds)
QUASI means a quasi-steady state
STEDY means a steady state

(4) After a QUASI state is reported, a print-out of the following format would be found

TIMERS
RE(ST) .
WAITT ...

TIMERS means 'data on the TDR's which are in the timing-state at this moment when a quasi-steady state has been reached' RE(ST) means the TDR edge number (its present state)

WAITT means the waiting time of the TDR

For example in the Test Circuit No 2 we have at RUNT = 10 0

TIMERS

RE(ST)	3(8)	8(8)
WAITT	0 5	1 0

This means that at this moment the TDR's whose edge Nos. are 3 and 8 are in the timing states 8 and 8 respectively and their waiting times are 0 5 and 1 0 respectively

(5) A print out of the form

*) RELAY TIME OUT (EJ=) No No

means the TDR's whose edge numbers are no , no , . ., timed out at this moment to pick up or drop out

(6) It should be remembered that a relay can have a total of 6 states (= 0,1,2,3,8 and 9) For the explanation of these states refer to Section 2 5 1

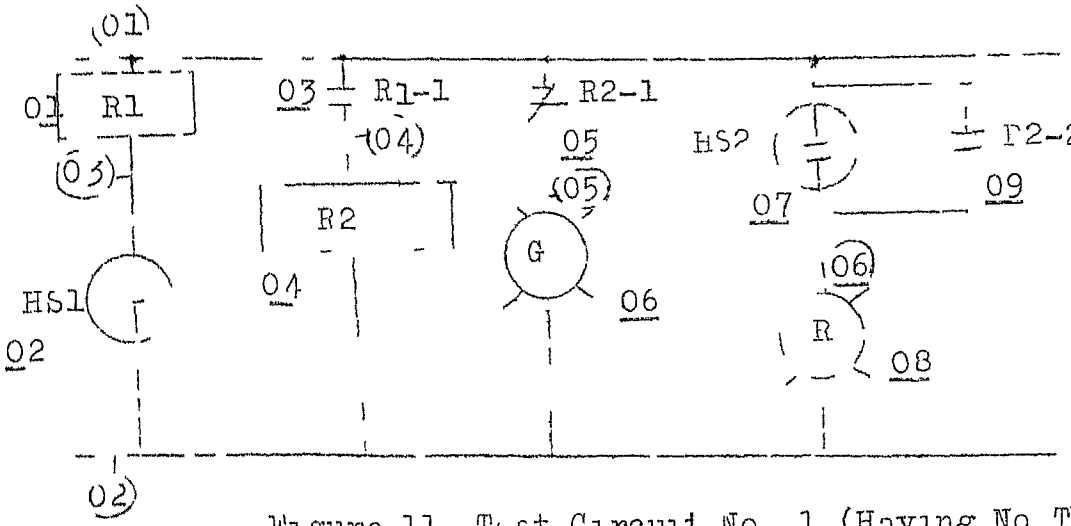


Figure 11 Test Circuit No 1 (Having No TDR's)

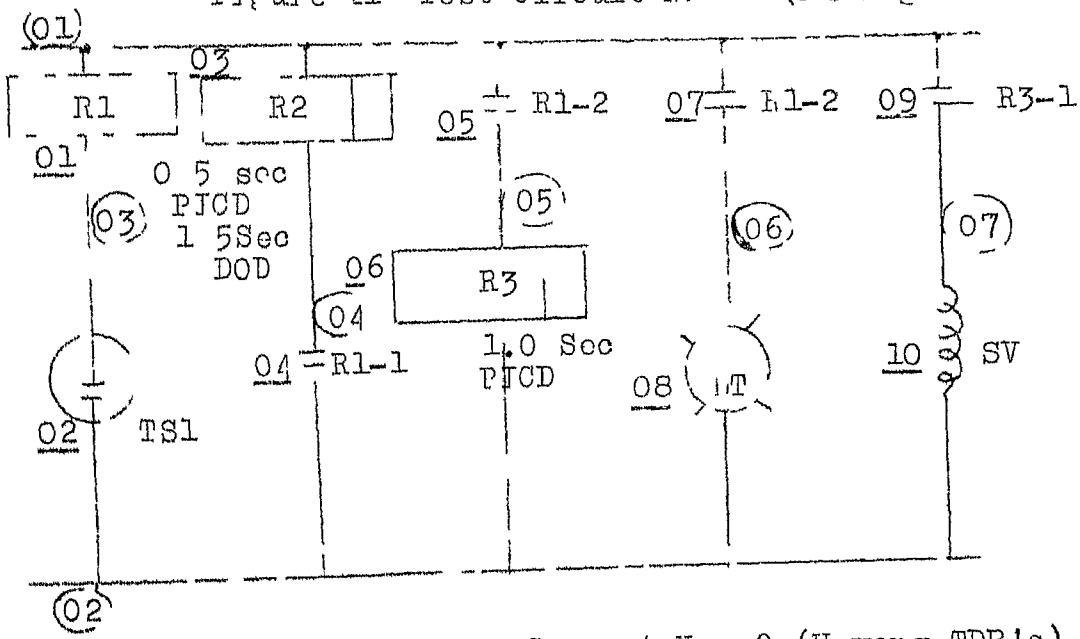


Figure 12 Test Circuit No 2 (Having TDR's)

Note The underlined numbers are the edge numbers, whereas the number enclosed in circles are the vertex numbers allotted.

Resistors Resistors are processed by the program as follows

(1) If between two vertices V1 and V2 there are only the resistors, then they are replaced by a connecting wire (i e , V1 and V2 are jumpered)

(2) All such resistors which have parallel to them an element other than resistors, are considered as non-existent (i e , having ∞ resistance).

Study of Fault Conditions in the Circuit Fault conditions can be easily studied by adding an input contact simulating the fault Occurrence of the fault is then simulated by closing or opening of such 'fault' contacts

(a) Accidental shorting of two points Add in the circuit an input contact, say FS, between the two points and keep its state = 0 Occurrence of the fault is simulated by setting the state of FS to 1 as an input event

(b) Open circuited element This can be simulated by adding an input contact FS in series with one of the terminals of the element Under no fault conditions, keep its state = 1 Occurrence of fault is specified by setting its state = 0.

TEST CIRCUIT NO 1 (HAVING NO TDR'S)

```
*1END
*2P
*3P
*4P
*5NØ
*6Λ
*7END
01,01,03,21,00/R1
02,03,02,11,00/HS1
03,01,04,01,01/R1-1
04,04,02,21,00/R2
05,01,05,02,04/R2-1
06,05,02,31,00/G
07,01,06,11,00/HS2
08,06,02,31,00/R
09,01,06,01,04/R2-2
99*
01,04
INITIAL INPUTS
02=0,07=0
THEN HS2=1
07=1
THEN HS2=0
07=0
THEN HS1 = 1
02=1
*8END
```

Data Cards for the Test Circuit No 1

TEST CIRCUIT NO. 2 (CIRCUIT HAVING TDR'S)

```
*1END
*2P
*3P
*4NØ
*5D
*6A
*7END
01,01,03,21,00/R1
02,03,02,11,00/TS
03,01,04,21,00/R2
04,04,02,01,01/R1-1
05,01,05,01,03/R2-1
06,05,02,31,00/LT
07,01,06,01,01/R1-2
08,06,02,21,00/R3
09,01,07,01,08/R3-1
10,07,02,31,00/SV
99*
03½000 5½001 5
08½001 0½000 0
99½000 0½000 0
01,03,08
INITIAL INPUTS
000 0
02=0
THEN TS CLOSED
010 0
02=1
TS OPENS BACK
000.3
02=0
CLOSES AGAIN
020 0
02=1
OPENS LATER
005 0
02=0
*8END
```

Data Cards for the Test Circuit No 2

REFERENCES

1 Caldwell, Samuel, H (1960) 'Switching Circuits and Logical Design', Wiley, New York

2 Hill, F J , and Peterson, G R , (1968), 'Introduction to Switching Theory and Logic Design', Wiley, New York

3 Deo, Narsingh (1974) 'Graph Theory with Applications to Engineering and Computer Science, Prentice-Hall, Inc

4 Knuth, Donald, E (1968) 'The Art of Computer Programming - Vol 1', Addison-Wesley, Reading, Mass

5 Kohavi, Zvi (1970) 'Switching and Finite Automata Theory, Tata McGraw-Hill, Bombay

6 Miller, Raymond, E (1965) 'Switching Theory, Vol 1', Wiley, New York

-

TEST CKT NO.1(HAVING NO TDR,S)

LIST OF ELEMENTS

1$ 1, $2 $ /R
2$ 3, $1 $ /HS
3$ 1, 4$ $ /R -1
4$ 4, $2 $ /R
5$ 1, $ 2$ 4/R2-1
6$ 5, $3 $ /G
7$ 1, 5$1 $ /HS2
8$ 6, $3 $ /R
9$ 1, $ $ 4/R2-2

DIAGRAM INFO PART- A
(A ZERO MEANS NIL)

EDGE	1	2	3	4	5	6	7	8	9
PARALLEL	0	0	0	0	0	0	9	0	0

DIAGRAM INFO PART- B

VERTEX LOWEST EDGE LEADING TO AN ADJACENT VERTEX

3 —	1	2	0	0	0	0	0	0
2 —	3	4	0	0	0	0	0	0
1 —	5	6	0	0	0	0	0	0
0 —	7	8	0	0	0	0	0	0

ANALYSIS

IN			OUT			
2	7	STATE	6	8	1	4

1) INITIAL INPUTS
0 0 TRANS 1 0 0 0

DIAGNOSIS
EDGE= 1 3 4 5 9
STATUS= 0 0 0 1 0
STEDY 1 0 0 0

DIAGNOSIS
EDGE= 1 3 4 5 9
STATUS= 0 0 0 1 0

2) THEN HS2=1
STEDY 1 1 0 0

DIAGNOSIS
EDGE= 1 3 4 5 9
STATUS= 0 0 0 1 0

3) THEN HS2=0
STEDY 1 0 0 0

DIAGNOSIS
EDGE= 1 3 4 5 9
STATUS= 0 0 0 1 0

4) THEN HS1=1
TRANS 1 0 0 0

DIAGNOSIS
EDGE= 1 3 4 5 9
STATUS= 1 1 0 1 0

TRANS 0 1 0 1

DIAGNOSIS
EDGE= 1 3 4 5 9
STATUS= 1 1 1 0 1

STEDY 0 1 1 1

DIAGNOSIS
EDGE= 1 3 4 5 9
STATUS= 1 1 1 0 1

//END OF ANALYSIS

TEST CKT NO. 2 (CKT HAVING 2 DR'S)

LIST OF ELEMENTS

[illegible]

DATA ON TIME DELAY RELAYS

COIL(PDG)	[illegible]	[illegible]	[illegible]
PICK-DELAY	[illegible]	[illegible]	[illegible]
DROP-DELAY	[illegible]	[illegible]	[illegible]

DIAGRAM INFO PART- A
(A ZERO MEANS NIL)

EDGE 1 2 3 4 5 6 7 8 9
PARALLEL 0 0 0 0 0 0 0 0 0

DIAGRAM INFO PART- B

VERT'X LOWEST EDGE LEADING TO AN ADJACENT VERTEX

1 --- 1 2 3 4 5 6 7
2 --- 3 5 6 8 9 1 4
3 --- 2 4 7 9 1 5 6
4 --- 7 8 2 6 3 9 1
5 --- 9 1 4 7 2 8 3

ANALYSIS

IN OUT

EN RUNTIME TLIN S STATE 1 2 3 8

1) INITIAL INPUTS
0.0 0.0 TRANS 1 2 3 8 8 ← line 2
STEDY 0 0 0 0 0

2) THEN TS CLOSED
10.0 20.0 TRANS 0 0 1 0 0
QUASI 0 0 1 8 8

TIMERS
RE(ST) 3(8) 8(8)
WAITT 0.5 1.0

TIMERS
RE(ST) 3(8) 8(8)
WAITT 0.5 1.0

3) TS OPENS BA+CK
10.3 0.3 TRANS 0 0 0 8 8
STEDY 0 0 0 0 0

4) CLOSES AGAIN
30.3 20.0 TRANS 0 0 1 0 0
QUASI 0 0 1 8 8

TIMERS
RE(ST) 3(8) 8(8)
WAITT 0.5 1.0

TIMERS
RE(ST) 3(8) 8(8)
WAITT 0.5 1.0

*) RELAY TIME-OUT (FU=)

TIMERS
RE(ST) 0(0)
WAITT 0

*) RELAY TIME-OUT (SEJ=)
TRANS
STEADY

F) OPENS LATER
TRANS
TRANS
QUASI

TIMERS
RE(ST) 0(9)
WAITT 0

TIMERS
RE(ST) 0(9)
WAITT 0

*) RELAY TIME-OUT (SEJ=)
TRANS
STEADY

// END OF ANALYSIS //

```
      INTEGER COILEJ,DIAGN
      INTEGER COMENT(16),OPT(6),COCOIL(99),EJ,NAME1(99),NAME2(99),
     1 CN,VA,VB,TEMP,EKA2,EKB1,PUDN,SN,MJDB(99),PLEJ(30),MARK(99),PREJ(30)
      INTEGER V1(99),V2(99),TYPE(99),IDA(16),COILES(18),VACS(18),
     1 VBCS(18),VADDS(16),VBODS(16),KOUKT(18),KTE(26,12),
     2 EJST(99),EXCASE(10),ADD(20),NEBAR(99,6),
     3 DIAGN,OPA(16),PRLEJ(99),QPED,QPSD,VALQ(30),
     4 EVNTEJ(16),EVNTST(16),KOUSC(18),PRIOCL(20),KOUSER(18),
     5 INAST(16),OPAST(16),PRST(13)
      DIMENSION PUD(13),DOD(18),WAITT(20)
      COMMON V1,V2,TYPE,INA,COILES,VACS,VBCS,VADDS,VBODS,KOUKT,KTE,
     1 EJST,EXCASE,ADD,NEBAR,DIAGN,PUD,DOD,KOUIN,KOUOP,KOUOD,KOUC,MAXEJ,
     2 MAXV,OPA,PRLEJ,QPED,QPSD,VALQ,EVNTEJ,EVNTST,KOUSC,NEVE,PRIOCL,
     3 KOUSER,WAITT,INAST,OPAST,PRST,JPOWER
      DATA ICH1/5H*1END/,ICH2/5H*2P/,ICH3/3H*3P/,ICH4/3H*4P/
      DATA ICH5/3H*5D/,ICH6/3H*6A/,ICH7/3H*7E/
C
C    SKIP TO A NEW PAGE
11    PRINT5
5     FORMAT(1H1,10X)
C
C    ... READ & PRINT INITIAL COMMENTS ...
C
11111 READ1,(COMENT(I),I=1,16)
1     FORMAT(16A5)
      IF(COMENT(1).EQ.ICH1) GOTO 13
      PRINT2,(COMENT(I),I=1,16)
2     FORMAT(10X,16A5)
      GOTO 11111
C     // END OF INITIAL COMMENTS //
13    CONTINUE
C    ... READ OPTIONS ...
      DO3 K=1,6
      READ4,OPT(K)
4     FORMAT(A5)
3     CONTINUE
C NOW CHECK IF ALL THE OPTION-CARDS WERE PLACED
      IF(OPT(6).EQ.ICH7) GOTO 9
      PRINT6
6     FORMAT(//15X,*ERROR--- OPTION CARDS NOT PROPER.*,
     1 * PROCESSING DELETED*).
      STOP
C
C     // END OF OPTIONS //
9     CONTINUE
      DIAGN=0
      IF(OPT(3).EQ.ICH4) DIAGN=1
C
C    ... READ ELEMENTS ...
C
      DO7 K=1,99
      V1(K)=0
      V2(K)=0
      TYPE(K)=0
      COCOIL(K)=0
7     CONTINUE
      MAXEJ=0
      MAXV=0
```

```
8       READ(3,*) J,V1(EJ),V2(EJ),TYPE(EJ),COCOIL(EJ),NAME1(EJ),NAME2(EJ)
10      FORMAT(I2,1X),2A4)
        IF(EJ.EQ.99) GOTO 16
        IF(EJ.GT.MAXEJ) MAXEJ=EJ
        IF(V1(EJ).GT.MAXV) MAXV=V1(EJ)
        IF(V2(EJ).GT.MAXV) MAXV=V2(EJ)
        GOTO 8
16      CONTINUE
C
C     // END OF ELEMENTS READING //
C
C    ...PREPARATION OF VARIOUS TABLES,ETC ...
C
        KOUC=0
        KOUIN=0
        KOUOD=0
        DO54 EJ=1,MAXEJ
        IF(TYPE(EJ).EQ.11) GOTO 56
        IF(TYPE(EJ).EQ.21) GOTO 58
        IF(TYPE(EJ).EQ.31) GOTO 60
        GOTO 54
56      KOUIN=KOUIN+1
        INA(KOUIN)=EJ
        GOTO 54
58      KOUC=KOUC+1
        COILES(KOUC)=EJ
        VACS(KOUC)=V1(EJ)
        VBCS(KOUC)=V2(EJ)
        GOTO 54
60      KOUOD=KOUOD+1
        OPA(KOUOD)=EJ
        VAODS(KOUOD)=V1(EJ)
        VBODS(KOUOD)=V2(EJ)
54      CONTINUE
C  **** RELATE COIL AND CONTACTS
C
        DO509 CN=1,KOUC
509     KOUKT(CN)=0
        DO201 EJ=1,MAXEJ
        IF(COCOIL(EJ).EQ.0) GOTO 201
C   ELSE THIS EJ IS A CONTACY. FIND RELAY SN TO WHICH IT BELONDS
C
        DO202 CN=1,KOUC
        IF(COILES(CN).NE.COCOIL(EJ)) GOTO 202
         KOUKT(CN)=KOUKT(CN)+1
        KTN=KOUKT(CN)
        KTE(CN,KTN)=EJ
202     CONTINUE
201     CONTINUE
C
C     // END OF TABLES ETC PREPARATION //
C
C    ... DETERMINATION OF PRL(EJ) ' NEBAR(JV,NEBN) ...
C
        DO20 J=1,MAXEJ
        PRLEJ(J)=0
        MARK(J)=0
20      CONTINUE
```

```
      DO32 LDEJ=1,MAXEJ
      IF((V1(LDEJ).EQ.0).OR.(MARK(LDEJ).EQ.1)) GOTO32
      MARK(LDEJ)=1
      LV1=V1(LDEJ)
      LV2=V2(LDEJ)
      JTE=LDEJ+1
      JUS=LDEJ
      DO34 K=JTE,MAXEJ
      KV1=V1(K)
      KV2=V2(K)
      IF(((KV1.EQ.LV1).AND.(KV2.EQ.LV2)).OR.
     1((KV1.EQ.LV2).AN).(KV2.EQ.LV1))) GOTO 36
      GOTO 34
C     K IS PARALLEL TO LDEJ
17    79 7 ++CA=8 +
      MARK(K)=1
      JUS=K
34    CONTINUE
32    CONTINUE
C END OF PARALLEL. NOW FIND NEBAR EDGES OF EACH VERTEX
      DO 38 J=1,MAXEJ
      MARK(J)=0
      MJOB(J)=0
38    CONTINUE
      DO40 J=1,MAXV
      DO42 K=1,8
      NEBAR(J,K)=0
42    CONTINUE
40    CONTINUE
      DO44 JV=3,MAXV
      NEBN=0
      DO46 JE=1,MAXEJ
      IF(MARK(JE).EQ.1) GOTO 46
      IF((V1(JE).NE.JV).AND.(V2(JE).NE.JV)) GOTO 46
C     ELSE JE IS A NEBAR OF JV
      NEBN=NEBN+1
      IF(NEBN.LE.8) GOTO 47
      PRINT48
48     FORMAT(15X,*ERROR--- VERTEX NO. *,I2,*HAS MORE*,
     1 *THAN 8 NEIGHBOJRING VERTICES. PROCESSING DELETED*)
      STOP
47    NEBAR(JV,NEBN)=JE
C NOW INVALIDATE PARALLELS IF NOT DONE ALREADY
      IF(MJOB(JE).EQ.1) GOTO 46
      MJOB(JE)=1
      JPE=PRLEJ(JE)
      DO50 K=1,MAXEJ
      IF(JPE.EQ.0) GOTO 46
      MARK(JPE)=1
      JPE=PRLEJ(JPE)
50    CONTINUE
46    CONTINUE
44    CONTINUE
C
C     // END OF PRL ! NEBAR DETERMINATION //
C
C    ... READ IOR DATA,IF ANY, AND ALLOT TO PROPER COIL NO.S ..
C
```

```
      IF(OPT(4).NE.ICH5) GOTO 250
C ELSE THERE ARE TDRS
      DO210 CN=1,18
      PUD(CN)=0.0
      DOD(CN)=0.0
210   CONTINUE
211   READ212,EJ,PICD,DROPD
212   FORMAT(I2,1X,F5.1,1X,F5.1)
      IF(EJ.EQ.99) GOTO 250
C  ELSE SEARCH COIL NO. OF THIS EJ AND RECORD DELAYS
      DO 214 CN=1,KOUC
      IF(COILES(CN).NE.EJ) GOTO 214
      PUD(CN)=PICD
      DOD(CN)=DROPD
214   CONTINUE
      GOTO 211
250   CONTINUE
C
C     /// END OF TDR DATA READING AND ALLOTMENT //
C
C    ... PROCESS DIAGRAM PRINTOUT OPTIONS ...
C          THESE ARE
C             1) LIST OF ELEMENTS
C             2) TDR DATA
C             3) STRUCTURE (TOPOLOGY)
C
      IF(OPT(1).NE.ICH2) GOTO 29
C ELSE PRINT LIST OF ELEMENTS
      PRINT 19
19    FORMAT(////45X,*LIST OF ELEMENTS*/)
      DO 20 J=1,MAKEJ
C SKIP UNUSED EJ NUMBERS
      IF(V1(J).EQ.0) GOTO 20
      PRINT21,J,V1(J),V2(J),TYPE(J),COCOIL(J),NAME1(J),NAME2(J)
21    FORMAT(40X,I2,1H$,I2,1H,,I2,1H$,I2,1H$,I2,1H/,2A4)
20    CONTINUE
C ALSO PRINT TDR DATA IF ANY
      IF(OPT(4).NE.ICH5) GOTO 29
      PRINT22
22    FORMAT(5X,*DATA ON TIME DELAY RELAYS*/)
      PRINT23,(COILES(CN),CN=1,KOUC)
      PRINT24,(PUD(CN),CN=1,KOUC)
23    FORMAT(1X,*COIL(EDGE)  *,17(I2,4X),I2)
24    FORMAT(1X,*PICK-DELAY *,18(F5.1,1X))
      PRINT 25,(DOD(CN),CN=1,KOUC)
25    FORMAT(1X,*DROP-DELAY *,18(F5.1,1X))
29    CONTINUE
C
C
      IF(OPT(2).NE.ICH3) GOTO 501
C ELSE PRINT INFO
      PRINT6000
6000  FORMAT(1H1,15X,*DIAGRAM INFO PART- A*/
     1 18X,*(A ZERO MEANS NIL)*)
      K=0
      DO90 EJ=1,MAKEJ
C     TYPE(EJ)=0 INDICATES A NON-EXSISTENT EDGE
C  PEJ IS EJ ARRAY ' PREJ IS PRL(EJ) ARRAY (MAX 30 FOR A LINE )
```

```
      IF(TYPE(EJ).EQ.2) GOTO 90
C     ELSE ENTER IT IN THE ARRAY
      K=K+1
      PEJ(K)=EJ
      PREJ(K)=PRLEJ(EJ)
      IF(K.LT.30) GOTO 77
C     ELSE THE ARRAY IS FULL . PRINT IT
      PRINT72,(PEJ(J),PREJ(J),J=1,30)
72    FORMAT(5X,*EDGE*,5X,30(I2,1X)/5X,*PARALLEL*,
     1 30(1X,I2)/)
C     RESET THE ARRAY POINTER TO ZERO
      K=0
      GOTO 90
C     SEE IF ALL THE EDGES ARE DONE, ALTHOUGH ARRAY NOT FULL. IF
C       YES - THEN PRINT ARRAY IF THERE IS ANYTHING COLLECTED IN IT
77    IF(EJ.LT.MAXEJ) GOTO 90
      PRINT74,(PEJ(J),J=1,K)
74    FORMAT(5X,*EDGE*,5X,30(I2,1X))
      PRINT75,(PREJ(J),J=1,K)
75    FORMAT(5X,*PARALLEL*,30(1X,I2))
90    CONTINUE
C  *** EJ VS PRL PRINTOUT COMPLLETE.  ***
C *** NOW PRINT PART-B OF DIAG INFO
101    PRINT102
102   FORMAT(///15X,*DIAGRAM INFO PART- B*//
     1 10X,*VERTEX  LOWEST EDGE LEADING TO AN ADJACENT VERTEX*/)
      DO104 JV=3,MAXV
C  SKIP UNUSED VERTEX NOS
      IF(NEBAR(JV,1).EQ.0) GOTO 104
      PRINT106,JV,(NEBAR(JV,K),K=1,8)
106   FORMAT(13X,I2,10(,*--*,8(1X,I2))
104   CONTINUE
C
C    // END OF DIAGRAM PRINTOUTS //
C
301   CONTINUE
C
C   ... INTERCHANGE,IF NECESSARY, VA ' VB OF COILS ' OD,S AND -
C         DETERMINE EXTENSION CASES FOR COILS ...
C
C  SET ALL ELEMENTS IN STATUS 1
      DO222 EJ=1,MAXEJ
      EJST(EJ)=1
222   CONTINUE
C   FIRST SETTLE COILS
      DO224 CN=1,KOUC
      VA=VACS(CN)
      VB=VBCS(CN)
      CALL PATH(VA,01,L1)
      CALL PATH(VB,02,L2)
      IF((L1*L2).EQ.1) GOTO 223
C  ELSE INTERCHANGE VACS ' VBCS
      TEMP=VACS(CN)
      VACS(CN)=VBCS(CN)
      VBCS(CN)=TEMP
C  THEN DETERMINE EXCASE(FOR COILS ONLY. OD,S JUST HANG)
      VA=VACS(CN)
      VB=VBCS(CN)
```

```
C  ABOVE IS DONE SINCE VA , VB MIGHT HAVE CHANGED
223     CALL PATH(VA,02,EXA2)
        CALL PATH(VB,01,EXB1)
        IF((EXA2.EQ.0).AND.(EXB1.EQ.0)) GOTO 226
        IF((EXA2.EQ.1).AND.(EXB1.EQ.0)) GOTO 228
        IF((EXA2.EQ.0).AND.(EXB1.EQ.1)) GOTO 230
        EXCASE(CN)=4
        GOTO 224
226     EXCASE(CN)=1
        GOTO 224
228     PRINT229,COILES(CN)
229     FORMAT(//35X,*CAUTION--- SHORT CKT CURRENT LIMITING RESISTOR*,
     1 * REQIRED IN +VE LEG OF COIL(EJ = *,I2,* )*)
        EXCASE(CN)=2
        GOTO 224
230     PRINT231,COILES(CN)
231     FORMAT(//35X,*CAUTION--- SHORT CKT CURRENT LIMITING RESISTOR *,
     1 *REQIRED IN -VE LEG OFCOIL(EJ = *,I2,* )*)
        EXCASE(CN)=3
224     CONTINUE
C  *** NOW SETTLE ODS
        DO232 ODN=1,KOUDD
        VA=VAODS(ODN)
        VB=VBODS(ODN)
        CALL PATH(VA,01,L1)
        CALL PATH(VB,02,L2)
        IF((L1*L2).EQ.1) GOTO 232
        TEMP=VAODS(ODN)
        VAODS(ODN)=VBODS(ODN)
        VBODS(ODN)=TEMP
232     CONTINUE
C
C    // END OF VA,VB INTERCHANGE AND DETERM OF EXCASE //
C
C    ... NOW PREPARE OP ARRAY ...
C
        IF(OPT(5).NE.IC46) GOTO 267
C   ELSE THERE ARE ADDITIONAL OP,S
        READ260,(ADO(J),J=1,15)
260     FORMAT(20(I2,1X))
C   THEN PUT THE ADD,S IN ARRAY OPA
        NEXT=KOUOD+1
        K=0
        DO265 SN=NEXT,15
        K=K+1
        IF(ADO(K).EQ.0) GOTO 265
        OPA(SN)=ADO(K)
263     CONTINUE
        KOUOP=16
        GOTO 268
265     KOUOP=SN-1
        GOTO 268
267     KOUOP=KOUOD
268     CONTINUE
C
C    // END OF OP ARRAY PREPARATION //
C
C    ... START OF ANALYSIS ...
```

```
C
C
C   PUT THE SYSTEM IN NO POWER STAT
C
      DO 302 CN=1,KOUC
      COILEJ=COILES(CN)
      EJST(COILEJ)=0
      PRST(CN)=0
      CALL KFS(CN,0)
302   CONTINUE
      DO 303 OPN=1,KOUOP
      OPAST(OPN)=0
      EJ=OPA(OPN)
      EJST(EJ)=0
303   CONTINUE
C     ANALYSIS IS DONE ACCORDING TO WHETHER
C      THERE ARE OR NOT ANY TDR,S
C
      JPOWER=1
      IF(OPT(4).EQ.IC+5) GOTO 351
C     ELSE THERE+ARE NO TDR,S
      CALL ANLYS1(JNEXT)
      IF(JNEXT.EQ.1) GOTO 11
      STOP
351   CALL ANLYS2(JNEXT)
      IF(JNEXT.EQ.1) GOTO 11
      STOP
      END
$IBFTC JPPEDA
      SUBROUTINE ANLYS1(JNEXT)
C
      INTEGER FMA(5),FMB(9),FMC(3),FNFT(10),FNST(8),
     1 FNFS(10),FNSS(8)
      INTEGER CN,REPORT,STATE,EVCOM(16),FINISH
      INTEGER V1(99),V2(99),TYPE(99),INA(16),COILES(18),VACS(18),
     1 VBCS(18),VAODS(16),VBODS(16),KOUKT(18),KTE(18,12),
     2 EJST(99),EXCASE(18),ADD(20),NEBAR(99,8),
     3 DIAGN,OPA(16),PRLEJ(99),QPED,QPSD,VALQ(30),
     4 EVNTEJ(16),EVNTST(16),KOUSC(18),PRIOCL(20),KOUSER(18),
     5 INAST(16),OPAST(16),PRST(18)
      DIMENSION PUD(18),DOD(16),WAITT(20)
      COMMON V1,V2,TYPE,INA,COILES,VACS,VBCS,VAODS,VBODS,KOUKT,KTE,
     1 EJST,EXCASE,ADD,NEBAR,DIAGN,PUD,DOD,KOUIN,KOUOP,KOUOD,KOUC,MAXEJ,
     2 MAXV,OPA,PRLEJ,QPED,QPSD,VALQ,EVNTEJ,EVNTST,KOUSC,NEVE,PRIOCL,
     3 KOUSER,WAITT,INAST,OPAST,PRST,JPOWER
      DATA FINISH/5H*END/,NEXTCK/5H*NEXT/
      DATA FMA/39H(5X,2HIN,          X,3HOUT/)    /,FMB
     1/54H(  5X,      (I2,1X),8H STATE  ,          (I2,1X))    /
      DATA FMC/18H(  5X,      (1H ))/,FNFT
     1/50H(  5X,      (1X,I1,1X),    8H TRANS  ,          (1X,I1,1X)) /
      DATA FNST/48H    (      X,  7HTRANS  ,          (1X,I1,1X)) /
      DATA JSTEDY/6H STEDY/,KSTEDY/6HSTEDY /
C     CONSTRUCTION OF PRINT FORMATS
      KI=KOUIN
      KO=KOUOP
      KA=3*KI+1
      KAB=3*(KI+KO)+8
      KC=3*KI+6
```

```
      WRITE(99,11)KA,KI,KO,KAB,KI,KO,KC,KO
11    FORMAT(8I6)
      READ(99,12)FMA(3),FMB(2),FMB(7),FMC(2),FNFT(2),FNFT(8),
     1 FNST(2),FNST(6)
12    FORMAT(8A6)
C       GENERATE FNS FROM FNFT
      WRITE(99,15)(FNFT(J),J=1,5),JSTEDY,(FNFT(K),K=7,10)
15    FORMAT(10A6)
      READ(99,15)(FNFS(J),J=1,10)
C       SIMILARLY GENERATE FNFS FROM FNST
      WRITE(99,17)(FNST(J),J=1,3),KSTEDY,(FNST(K),K=5,8)
17    FORMAT(8A6)
      READ(99,17)(FNSS(J),J=1,8)
C       PRINT ANALYSIS HEADINGS
      PRINT32
32    FORMAT(////25X,*ANALYSIS*/)
      WRITE(6,FMA)
      WRITE(6,FMB)(INA(J),J=1,KI),(OPA(K),K=1,KO)
      WRITE(6,FMC)
      NEVE=0
      DO45 CN=1,18
      DOD(CN)=0.0
      PUD(CN)=0.0
45    CONTINUE
      PRIOCL(1)=0
51    READ55,(EVCOM(I),I=1,16)
55    FORMAT(16A5)
      IF(EVCOM(1).NE.FINISH) GOTO 60
      PRINT58
58    FORMAT(//30X,*// END OF ANALYSIS //*)
      STOP
60    IF(EVCOM(1).NE.NEXTCK) GOTO 65
      PRINT58
C     PROCESS NEXT CKT
      JNEXT=1
      RETURN
65    NEVE=NEVE+1
      PRINT68,NEVE,(EVCOM(I),I=1,16)
68    FORMAT(2X,I2,*) *,16A5)
C     THEN READ THE INPUT EVENT DATA
      READ71,(EVNTEJ(J),EVNTST(J),J=1,16)
71    FORMAT(16(I2,1X,I1,1X))
C     THEN PLANT THIS EVENT IN THE SYSTEM
      CALL UPISTA
C     INITIALIZATIONS FOR THE SCAN
      DO89 CN=1,KOUC
89    KOUSC(CN)=0
      REPORT=1
91    CALL SCANCL(STATE)
C     THEN CHECK WHETHER THIS IS A TRANS OR STEDY STATE
      IF(STATE.EQ.3) GOTO 101
C     ELSE IT IS ONLY A TRANS STATE(TRF-Q NOT YET EMPTY)
      GOTO(94,95),REPORT
C       FIRST REPORT FOR THE EVENT * STATE IS TRANS
94    WRITE(6,FNFT)(INAST(J),J=1,KI),(OPAST(K),K=1,KO)
      IF(DIAGN.EQ.1) CALL DIAGNS
      REPORT=2
      GOTO 91
```

```
C       PRINT SUBSEQENT REPORT FOR HOLD EVENT , STATE IS TRANS
95      WRITE(6,FNST)(OPAST(K),K=1,KO)
        IF(DIAGN.EQ.1) CALL DIAGNS
C       THEN DETERMINE THE NEXT STATE
        GOTO 91
C       STATE IS STEDY
101     GOTO(114,115),REPORT
114     WRITE(6,FNFS)(INAST(J),J=1,KI),(OPAST(K),K=1,KO)
        IF(DIAGN.EQ.1) CALL DIAGNS
C       ELSEGOTO READ NEXT EVENT
        GOTO 51
115     WRITE(6,FNSS)(OPAST(K),K=1,KO)
        IF(DIAGN.EQ.1) CALL DIAGNS
C       ELSE GOTO READ NEXT EVENT
        GOTO 51
        END
$IBFTC JPPEDB
        SUBROUTINE ANLYS2(JNEXT)
C
C.....
C       THIS SUB. DOES ANALYSIS OF THE CIRCUIT HAVING TDR,S
        INTEGER FMJ(5),FMK(12),FML(3),FEFT(13),FRFT(9),FST(8),
     1  FEFI(13),FEFS(13),FRFI(9),FRFS(9),FSI(8),FSS(8)
        INTEGER CN,REPORT,STATE,EVCOM(16),FINISH,SLOCYC
        INTEGER V1(99),V2(99),TYPE(99),INA(16),COILES(18),VACS(18),
     1  VBCS(18),VAODS(16),VBODS(16),KOUKT(18),KTE(18,12),
     2  EJST(99),EXCASE(18),ADD(20),NEBAR(99,8),
     3  DIAGN,OPA(16),PRLEJ(99),QPED,QPSD,VALQ(30),
     4  EVNTEJ(16),EVNTST(16),KOUSC(18),PRIOCL(20),KOUSER(18),
     5  INAST(16),OPAST(16),PRST(18)
        DIMENSION PUD(18),DOD(18),WAITT(20)
        COMMON V1,V2,TYPE,INA,COILES,VACS,VBCS,VAODS,VBODS,KOUKT,KTE,
     1  EJST,EXCASE,ADD,NEBAR,DIAGN,PUD,DOD,KOUIN,KOUOP,KOUOD,KOUC,MAXEJ,
     2  MAXV,OPA,PRLEJ,QPED,QPSD,VALQ,EVNTEJ,EVNTST,KOUSC,NEVE,PRIOCL,
     3  KOUSER,WAITT,INAST,OPAST,PRST,JPOWER
        DATA FINISH/5H*END/,NEXTCK/5H*NEXT/
        DATA JIMIDT/6HQJASI /,JSTEDY/6HSTEDY /
        DATA FMJ/30H( 18X,2HIN,        X,3HOUT/)    /,FMK
     1/72H(1H ,17HEN RUNTIME  TLIN ,          (I2,1X),6HSTATE ,        (IREL(
     22,1X))    /
        DATA FML/18H(1H ,        (1H )/,FEFT
     1/78H( 4X,F7.1, 1X, F5.1, 1X,      (1X,I1,1X),     6HTRANS ,
     2    (1X,I1,1X)) /
        DATA FRFT/54H(  4X, F7.1,       X,  6HTRANS ,           (1X,I1,1X)
     1) /
        DATA FST/48H     (       X,  6HTRANS ,          (1X,I1,1X)) /
        KI=KOUIN
        KO=KOUOP
        KJ=3*KI+2
        KJK=3*(KI+KO)+23
        KL=3*KI+7
        KM=18+3*KI
        WRITE(99,11)KJ,KI,KO,KJK,KI,KO,KL,KO,KM,KO
11      FORMAT(10I6)
        READ(99,12)FMJ(3),FMK(6),FMK(12),FML(2),
     1  FEFT(5),FEFT(11),FRFT(5),FRFT(7),FST(2),FST(6)
12      FORMAT(10A6)
C       GENERATE FEFI ' FEFS FROM FEFT
```

```
      WRITE(99,15)(FRFT(J),J=1,8),JIMIDT,(FRFT(K),K=10,13)
15    FORMAT(12A6)
      READ(99,15)(FRFI(J),J=1,12)
      WRITE(99,15)(FRFT(J),J=1,8),JSTEDY,(FRFT(K),K=10,13)
      READ(99,15)(FRFS(J),J=1,12)
C     LIKEWISE GENERATE FRFI ' FRFS FROM FRFT
      WRITE(99,15)(FRFT(J),J=1,4),JIMIDT,(FRFT(K),K=6,9)
      READ(99,15)(FRFI(J),J=1,9)
      WRITE(99,15)(FRFT(J),J=1,4),JSTEDY,(FRFT(K),K=6,9)
      READ(99,15)(FRFS(J),J=1,9)
C     GENERATE FSI ' FSS FROM FST
      WRITE(99,15)(FST(J),J=1,3),JIMIDT,(FST(K),K=5,8)
      READ(99,15)(FSI(J),J=1,8)
      WRITE(99,15)(FST(J),J=1,3),JSTEDY,(FST(K),K=5,8)
      READ(99,15)(FSS(J),J=1,8)
C       PRINT ANALYSIS HEADINGS
      PRINT32
32    FORMAT(////25X,*ANALYSIS*/)
      WRITE(6,FMJ)
      WRITE(6,FMK)(INA(J),J=1,KI),(OPA(K),K=1,KO)
      WRITE(6,FML)
C     INITIALIZATIONS FOR THE WAIT Q
      DO31 J=1,20
      PRIOCL(J)=0
      WAITT(J)=99999.0
31    CONTINUE
      RUNT=00000.0
      NEVE=0
51    READ55,(EVCOM(I),I=1,16)
55    FORMAT(16A5)
      IF(EVCOM(1).NE.FINISH) GOTO 60
      PRINT58
58    FORMAT(//30X,*// END OF ANALYSIS //*)
      STOP
60    IF(EVCOM(1).NE.NEXTCK) GOTO 65
      PRINT58
C     THEN PROCESS NEXT CKT
      JNEXT=1
      RETURN
65    NEVE=NEVE+1
      PRINT66,NEVE,(EVCOM(J),J=1,16)
68    FORMAT(//1H ,I2,*) *,16A5)
C     THEN READ TLI
      READ70,TLI
70    FORMAT(F5.1)
      RTE=TLI
C     THEN READ EVENT DATA
      READ71,(EVNTEJ(J),EVNTST(J),J=1,16)
71    FORMAT(16(I2,1X,I1,1X))
C     PLANT EVENT IN THE SYSTEM
72    CALL UPISTA
      RUNT=RUNT+RTE
      DO75 CN=1,KOUC
75    KOUSER(CN)=0
C       INITIALIZATIONS FOR THE SCAN
77    DO89 CN=1,KOUC
89    KOUSC(CN)=0
      REPORT=1
```

```
C       THEN D GET THE NEXT STATE
101     CALL SCANCL(STATS)
C       THEN FIND WHICH STATE IS REACHED
C           STATE=1 FOR TRANSIENT, =2 FOR QUASI '=3 FOR STEADY STATE
        IF(STATE.EQ.1) GOTO 121
        IF(STATE.EQ.2) GOTO 141
C       ELSE STATE IS 3 (STEDY STATE ON INPUT EVENT)
        GOTO(104,114),REPORT
104     WRITE(6,FEFS)RUNT,TLI,(INAST(J),J=1,KI),(OPAST(K),K=1,KO)
        IF(DIAGN.EQ.1) CALL DIAGNS
C       THEN GOTO READ NEXT EVENT
        GOTO 51
C       STEDY STATE IS REACHED FOR THIS INPUT EVENT ON A SUBSEQENT SCAN
114     WRITE(6,FSS)(OPAST(J),J=1,KO)
        IF(DIAGN.EQ.1) CALL DIAGNS
        GOTO 51
C       SO IT IS TRANS FOR THE INPUT EVENT
121     GOTO(124,134),REPORT
124     WRITE(6,FEFT)RUNT,TLI,(INAST(J),J=1,KI),(OPAST(K),K=1,KO)
        IF(DIAGN.EQ.1) CALL DIAGNS
        REPORT=2
        GOTO 101
134     WRITE(6,FST)(OPAST(K),K=1,KO)
        IF(DIAGN.EQ.1) CALL DIAGNS
C        DETERMINE NEXT STATE
        GOTO 101
C       A QUASI STEADY STATE HAS REACHED
141     GOTO(154,164),REPORT
154     WRITE(6,FEFI)RUNT,TLI,(INAST(J),J=1,KI),(OPAST(K),K=1,KO)
        IF(DIAGN.EQ.1) CALL DIAGNS
C       SINCE SOME TIMERS ARE IN WAIT-Q
        GOTO 171
164     WRITE(6,FSI)(OPAST(K),K=1,KO)
        IF(DIAGN.EQ.1) CALL DIAGNS
171      CONTINUE
C       NOW READ NEXT INPUT EVENT ' DETERMINE WHETHER THE INPUT OR THE
C           SWR SHOULD BE ATTENDED TO FIRST (SWR MEANS THE WAITNG RELAY
C           WITH SMALLEST WAITT IE HAVING HIGHEST PRIORITY)
        NEVE=NEVE+1
        READ55,(EVCOM(I),I=1,16)
C        CHECK IF EVENTS HAVE ENDED UP
        IF((EVCOM(1).NE.FINISH).AND.
     1 (EVCOM(1).NE.NE(TCK)) GOTO 201
C        ELSE IT IS FINISH OR NEXTCK
C      EVENTS HAVE ENDED UP BUT TIMERS ARE STILL WAIT-Q. THEY ALL MUST BE  SER
        RTE=999999.9
        GOTO 251
C        READ ALSO THE TLI ' INPUT DATA
201     READ70,TLI
        RTE=TLI
        READ71,(EVNTEJ(J),EVNTST(J),J=1,16)
251     CONTINUE
        CALL PWATRS
C       NOW CHECK WHETHER SWR OR THE INPUT READ ALREADY SHOULD BE ATTENDED TO
        SWT=WAITT(1)
        IF(SWT.GT.RTE) GOTO 2001
        IF(SWT.EQ.RTE) GOTO 2001
C       ELSE SWT IS LT RTE. HENCE SWR TO BE SERVED FIRST
```

```
      RUNT=RUNT+SWT
      RTE=RTE-SWT
C       INITIALIZATIONS FOR THE SCAN
201   DO289 CN=1,KOUC
289   KOUSC(CN)=0
      REPORT=1
      CALL SERADQ(SLOCYC)
      IF(SLOCYC.EQ.0) GOTO 301
C     ELSE SLOW CYCLING OCURRING. MESSAGE ALREADY PRINTED INSUB. SERADQ
      STOP
301    CALL SCANCL(STATE)
C     CHECK WHICH STATE REACHED
      IF(STATE.EQ.1) GOTO 351
      IF(STATE.EQ.2) GOTO 381
C     ELSE STATE=3(STEDY STATE)
C
C     NOTHING IN ANY Q AFTER UPDATING TDR
C
      GOTO(304,314),REPORT
C     STEDY STATE ON FIRST SCAN ITSELF
304   WRITE(6,FRFS)RUNT,(OPAST(K),K=1,KO)
      IF(DIAGN.EQ.1) CALL DIAGNS
      GOTO 320
314   WRITE(6,FSS)(OPAST(K),K=1,KO)
      IF(DIAGN.EQ.1) CALL DIAGNS
320   CONTINUE
C     NOW NATURALLY EW HAVE TO ATTEND TO THE INPUT EVENT READ ALREADY
      IF(EVCOM(1).NE.FINISH) GOTO 330
      PRINT58
      STOP
330   IF(EVCOM(1).NE.NEXTCK) GOTO 335
      PRINT58
      JNEXT=1
      RETURN
C
335   CONTINUE
C
C       SO WE HAVE TO SERVE THE EVENT READ ALREADY
      PRINT68,NEVE,(EVCOM(J),J=1,16)
      GOTO 72
C     TRANS STATE AFTER TDR UPDARING
351   GOTO(354,364),REPORT
354   WRITE(6,FRFT)RUNT,(OPAST(K),K=1,KO)
      IF(DIAGN.EQ.1) CALL DIAGNS
      REPORT=2
      GOTO 381
364   WRITE(6,FST)(OPAST(K),K=1,KO)
      IF(DIAGN.EQ.1) CALL DIAGNS
C
      GOTO 381
C       A QUASI STEDY STATE HAS REACHED
381   GOTO(384,394),REPORT
384   WRITE(6,FRFI)RUNT,(OPAST(K),K=1,KO)
      IF(DIAGN.EQ.1) CALL DIAGNS
      GOTO 397
394   WRITE(6,FSI)(OPAST(K),K=1,KO)
      IF(DIAGN.EQ.1) CALL DIAGNS
C
```

```
397     CONTINUE
C
C        QUASI STATE HAS BEEN PRINTED OUT. CHECK WHETHER
C              AT LEST NOW INPUT EVENT CAN BE ATTENDED TO
        GOTO 251
C
1001    CONTINUE
C       EVENT HAS BEEN READ AND IT IS FOUND THAT THIS EVENT SHOULD BE ATTENDE
C         TO FIRST THAN SWR
        PRINT68,NEVE,(EVCOM(J),J=1,16)
C       THEN REDUCE WAITT OF ALL THE WAITERS BY RTE
        DO1009 J=1,20
        IF(PRIOCL(J).EQ.0) GOTO 72
        WAITT(J)=WAITT(J)-RTE
1009    CONTINUE
        GOTO 72
C
2001    CONTINUE
C
C       INPUT EVENT HAS BEEN READ ' IT IS FOUND THAT SWR ' INPUT
C            SHOULD BE JPDATED SIMULTANROUSLY
C
        DO2009 J=1,KOUC
2009    KOUSER(J)=0
        CALL UPISTA
        CALL SERADQ(SLOCYC)
C        SLOCYC=1 RULED OUT HERE. HENCE NOT CHECKED
        PRINT2012
2012    FORMAT(6X,*SIMULTANEOUSLY*)
        PRINT2018,NEVE,(EVCOM(J),J=1,16)
2018    FORMAT(1H ,I2,*) *,16A5)
        RUNT=RUNT+RTE
C       THEN GO TO UPDATE THE SYSTEM TO NEXT STATE
        GOTO 77
        END
$IBFTC JPPEDD
        SUBROUTINE DIAGNS
C
C
        INTEGER EJ,PEJ(30),PST(30)
        INTEGER V1(99),V2(99),TYPE(99),INA(16),COILES(18),VACS(18),
      1 VBCS(18),VAODS(16),VBODS(16),KOUKT(18),KTE(18,12),
      3 DIAGN,OPA(16),PRLEJ(99),QPED,QPSD,VALQ(30),
      2 EJST(99),EXCASE(18),ADD(20),NEBAR(99,8),
      4 EVNTEJ(16),EVNTST(16),KOUSC(18),PRIOCL(20),KOUSER(18),
      5 INAST(16),OPAST(16),PRST(18)
        DIMENSION PUD(18),DOD(18),WAITT(20)
        COMMON V1,V2,TYPE,INA,COILES,VACS,VBCS,VAODS,VBODS,KOUKT,KTE,
      1 EJST,EXCASE,ADD,NEBAR,DIAGN,PUD,DOD,KOUIN,KOUOP,KOUOD,KOUC,MAXEJ,
      2 MAXV,OPA,PRLEJ,QPED,QPSD,VALQ,EVNTEJ,EVNTST,KOUSC,NEVE,PRIOCL,
      3 KOUSER,WAITT,INAST,OPAST,PRST,JPOWER
C    THIS SUBROURINE PRINTS OUTSTATUS OF ALL THE RELAY COILS AND
C     THEIR CONTACTS AT THE TIME OF SYSTEM STATUS REPORTEDLAST BY THE
C     MAIN PROGRAM. STATUS 0 OF INPUT CONTACTS ' OUTPUT DEVICES ALREADY
C      30 EDGES REPORYED IN A LINE
C
        K=
        DO19 EJ=1,MAXEJ
```

```
      JT=TYPE(EJ)
C     JT= MEANS A NON-EXISTENT EDGE NO.
      IF((JT.EQ.0).OR.(JT.EQ.11).OR.(JT.GE.21)) GOTO 7
C     ELSE ENTER IT IN THE ARRAY
      K=K+1
      PEJ(K)=EJ
      PST(K)=EJST(EJ)
      IF(K.LT.30) GOTO 7
C     ELSE THE ARRAY IS FULL. PRINT IT
      PRINT11,(PEJ(J),J=1,30)
11    FORMAT(/22X,*DIAGNOSIS*/22X,*EDGE= *,30(I2,1X))
      PRINT12,(PST(J),J=1,30)
12    FORMAT(20X,*STATUS= *,30(1X,I1,1X))
C     RESET THE ARRAY POINTER TO ZERO
      K=0
      GOTO 19
C
C     SEE IF ALL THE EDGES ARE DONE, EVENTHOUGH ARRAY NOT FULL. IF
C     YES THEN PRINT ARRAY IF THERE IS ANYTHING IN IT.
C
7     IF(EJ.LT.MAXEJ) GOTO 19
      IF(K.EQ.0) GOTO 19
      PRINT11,(PEJ(J),J=1,K)
      PRINT12,(PST(J),J=1,K)
19    CONTINUE
      RETURN
      END
$IBFTC JPPEDE
      SUBROUTINE ENTPRQ(CN,DELAY)
C
C     THIS SUB. ENTERS A COIL AT THE PROPER PLACE(POSITION) IN
C     THE PRIORITY QUEUE(WAIT-Q) . STRAIGHT INSERTION IS USED.
C
      INTEGER CN,QP
      INTEGER V1(99),V2(99),TYPE(99),INA(16),COILES(18),VACS(18),
     1 VBCS(18),VAODS(16),VBODS(16),KOUKT(18),KTE(18,12),
     2 EJST(99),EXCASE(18),ADD(20),NEBAR(99,8),
     3 DIAGN,OPA(16),PRLEJ(99),QPED,QPSD,VALQ(30),
     4 EVNTEJ(16),EVNTST(16),KOUSC(18),PRIOCL(20),KOUSER(18),
     5 INAST(16),OPAST(16),PRST(16)
      DIMENSION PUD(18),DDC(18),WAITT(20)
      COMMON V1,V2,TYPE,INA,COILES,VACS,VBCS,VAODS,VBODS,KOUKT,KTE,
     1 EJST,EXCASE,ADD,NEBAR,DIAGN,PUD,DDD,KOUIN,KOUOP,KOUOD,KOUC,MAXEJ,
     2 MAXV,OPA,PRLEJ,QPED,QPSD,VALQ,EVNTEJ,EVNTST,KOUSC,NEVE,PRIOCL,
     3 KOUSER,WAITT,INAST,OPAST,PRST,JPOWER
      QP=20
C     HIGHEST POSITION A COIL CAN BE IN Q IS 18
1     QP=QP-1
      IF(QP.EQ.0) GOTO 3
      IF(DELAY.GE.WAITT(QP)) GOTO 3
C     ELSE DELAY IS LT WAITT(QP). SHIFT PRIOCL(QP) ONE PLACE TO RIGHT
      PRIOCL(QP+1)=PRIOCL(QP)
      WAITT(QP+1)=WAITT(QP)
C     THEN COMPARE WITH THE NEXT LOWER POSITION
      GOTO 1
C     PUT CN TO THE RIGHT OF QP
3     PRIOCL(QP+1)=CN
      WAITT(QP+1)=DELAY
```

```
      RETURN
      END
$IBFTC JPP  DF
      SUBROUTINE KFS(CN,ST)
      INTEGER CN,ST,EJ
      INTEGER V1(99),V2(99),TYPE(99),INA(16),COILES(18),VACS(18),
     1 VBCS(18),VAODS(18),VBODS(16),KOUKT(18),KTE(18,12),
     2 EJST(99),EXCASE(18),ADD(20),NEBAR(99,8),
     3 DIAGN,OPA(16),PRLEJ(99),QPED,QPSD,VALQ(30),
     4 EVNTEJ(16),EVNTST(16),KOUSC(18),PRIOCL(20),KOUSER(18),
     5 INAST(16),OPAST(16),PRST(18)
      DIMENSION PUD(18),DOD(18),WAITT(20)
      COMMON V1,V2,TYPE,INA,COILES,VACS,VBCS,VAODS,VBODS,KOUKT,KTE,
     1 EJST,EXCASE,ADD,NEBAR,DIAGN,PUD,DOD,KOUIN,KOUOP,KOUOD,KOUC,MAXEJ,
     2 MAXV,OPA,PRLEJ,QPED,QPSD,VALQ,EVNTEJ,EVNTST,KOUSC,NEVE,PRIOCL,
     3 KOUSER,WAITT,INAST,OPAST,PRST,JPOWER
C       THIS SUB.SETS THE STATUS OF THE CONTACTS OF RELAY COIL CN
C          CORRESPONDING TO STATE ST OF THE COIL. TRANSFER
C             CONTACTS ARE ALSO SET TO THER FINAL(' NOT TRANSITORY) STATE.
      MAXKT=KOUKT(CN)
      IF(ST.EQ.1) GOTO 51
C   ELSE ST IS 0
C
      DO49 KTN=1,MAXKT
      EJ=KTE(CN,KTN)
      JT=TYPE(EJ)
      IF((JT.EQ.1).OR.(JT.EQ.3).OR.(JT.EQ.5)) GOTO 11
      EJST(EJ)=1
      GOTO 49
11    EJST(EJ)=0
49    CONTINUE
      RETURN
51    DO99 KTN=1,MAXKT
      EJ=KTE(CN,KTN)
      JT=TYPE(EJ)
       IF((JT.EQ.2).OR.(JT.EQ.4).OR.(JT.EQ.6)) GOTO 61
      EJST(EJ)=1
      GOTO 99
61    EJST(EJ)=0
99    CONTINUE
      RETURN
      END
$IBFTC JPPEDG
      SUBROUTINE PATH(SV,GV,LUCK)
C  THIS SUBROUTINE RETURNS LUCK=1 OR 0 ACCORDING AS TO WHETHER
C  THERE IS OR THERE IS NOT A TRANSMISSION(PATH) BETWEEN
C  VERTEX SV AND GV
C
      INTEGER SV,GV,MARK(99),VL,STP,VSTK(30),BNSTK(30),
     1 VX,BRNO,EJT,VA,VB,TP
      INTEGER V1(99),V2(99),TYPE(99),INA(16),COILES(18),VACS(18),
     1 VBCS(18),VAODS(18),VBODS(16),KOUKT(18),KTE(18,12),
     2 EJST(99),EXCASE(18),ADD(20),NEBAR(99,8),
     3 DIAGN,OPA(16),PRLEJ(99),QPED,QPSD,VALQ(30),
     4 EVNTEJ(16),EVNTST(16),KOUSC(18),PRIOCL(20),KOUSER(18),
     5 INAST(16),OPAST(16),PRST(18)
      DIMENSION PUD(18),DOD(18),WAITT(20)
      COMMON V1,V2,TYPE,INA,COILES,VACS,VBCS,VAODS,VBODS,KOUKT,KTE,
```

```
      1 EJST,CYCASE,ADD,NEBAR,DIAGN,POD,DOD,KOUIN,KOUOP,KOUOD,KOUC,MAXEJ,
      2 MAXV,OPL,PRLEJ,SPLD,OPSD,VALQ,VALTPJ,EVNTST,KOUSC,NEVE,PRIOCL,
      3 KOUSCR,WAITT,INAST,OPAST,PRST,JPOWER
C
C
      IF(SV.EQ.GV) GOTO 999
      DO1 J=1,MAXV
1     MARK(J)=
      VL=SV
      STP=1.
11    VSTK(STP)=VL
      MARK(VL)=1
      BNSTK(STP)=1
      VX=VSTK(STP)
6     BRNO=BNSTK(STP)
      EJT=NEBAR(VX,BRNO)
      IF(EJT.EQ.0) GOTO 101
      GOTO 103
101   STP=STP-1
      IF(STP.NE.0) GOTO 102
      LUCK=0
      RETURN
C
C  TRY NEXT BRANCH OF THE EARLIER VERTEX
102   VX=VSTK(STP)
      BNSTK(STP)=BNSTK(STP)+1
      GOTO 6
103   VA=V1(EJT)
      VB=V2(EJT)
      IF(VA.EQ.VX) GOTO 3
      NPV=VA
      GOTO 4
3     NPV=VB
C
C  NOW SEE IF WE CAN ACTUALLY EXTEND PATH TO NPV THRU EJT
C  1) NO USE EXTENDING PATH TO NPV IF IT HAS BEEN ALREADY
C     TRAVERSED(IE IS MARKED)
C  2) WE CANNOT TRAVERSE THEEDGE IF IT IS A COIL OR AN OUTPUT DEVICE
C  3) IF EJT IS A DIODE ' SV=01 THEN TRANSMISSION ALLOWED ONLY IF
C     VERT1(EJT)=NPV . IF SV=02 THEN TRANSMISSION POSSIBLE ONLY IF
C      VERT2(EJT)=NPV
4     IF(MARK(NPV).EQ.1) GOTO 5
      JRP=0
C        JRP=1 AS LONG AS ONLY RESISTORS ARE FOUND
      DO201 J=1,MAXEJ
      TP=TYPE(EJT)
      IF((TP.EQ.51).AND.(JRP.EQ.0)) GOTO 151
      IF((TP.EQ.51).AND.(JRP.EQ.1)) GOTO 195
       JRP=1
      IF((TP.EQ.21).OR.(TP.EQ.31)) GOTO 195
C  ELSE EJT IS A DIODE OR A CONTACT
C
      IF(TP.EQ.41) GOTO 191
C  ELSE EJT IS A CONTACT
C
      IF(EJST(EJT).EQ.1) GOTO 202
      GOTO 195
151   CONTINUE
```

```
151     CONTINUE
        EJT=PRLEJ(EJT)
        IF(EJT.EQ.0) GOTO 202
        GOTO 201
C
C  FOR DIODE
191     IF(((GV.EQ.01).AND.(VA.EQ.NPV)).OR.
     1  ((GV.EQ.02).AND.(VB.EQ.NPV))) GOTO 202
195     EJT=PRLEJ(EJT)
        IF( JT.EQ.0) GOTO 5
201     CONTINUE
202     VL=NPV
        IF(VL.EQ.GV) GOTO 999
        STP=STP+1
        GOTO 11
5       BNSTK(STP)=BNSTK(STP)+1
        GOTO 6
999     LUCK=1
        RETURN
        END
$IBFTC JPPEDH
        SUBROUTINE PWATRS
C       THIS SUB. PRINTS DATA ABOUT THE RELAYS WHICH ARE IN THE
C       WAITNG PHASE DUE THE PUD OR DOD
        INTEGER CN,EJ,PEJ(20),PST(20)
        INTEGER V1(99),V2(99),TYPE(99),INA(16),COILES(18),VACS(18),
     1 VBCS(18),VAODS(16),VBODS(16),KOUKT(18),KTE(18,12),
     2 EJST(99),EXCASE(18),ADD(20),NEBAR(99,8),
     3 DIAGN,OPA(16),PRLEJ(99),QPED,QPSD,VALQ(30),
     4 EVNTEJ(16),EVNTST(16),KOUSC(18),PRIOCL(20),KOUSER(18),
     5 INAST(16),OPAST(16),PRST(18)
        DIMENSION PUD(18),DOD(18),WAITT(20)
        COMMON V1,V2,TYPE,INA,COILES,VACS,VBCS,VAODS,VBODS,KOUKT,KTE,
     1 EJST,EXCASE,ADD,NEBAR,DIAGN,PUD,DOD,KOUIN,KOUOP,KOUOD,KOUC,MAXEJ,
     2 MAXV,OPA,PRLEJ,QPED,QPSD,VALQ,EVNTEJ,EVNTST,KOUSC,NEVE,PRIOCL,
     3 KOUSER,WAITT,INAST,OPAST,PRST,JPOWER
        DO99 J=1,20
        IF(PRIOCL(J).EQ.0) GOTO 121
C       PRIOCL(J)=0 BEFORE J=20. THUS THE VALUE OF J WILL
C         BE AVAILABLE OUTSIDE DO LOOP
        CN=PRIOCL(J)
        EJ=COILES(CN)
        PEJ(J)=EJ
        PST(J)=EJST(EJ)
99      CONTINUE
121     PRINT134
134     FORMAT(/9X,*TIMERS*)
        J=J-1
        PRINT147,(PEJ(K),PST(K),K=1,J)
147     FORMAT(4X,*RELAY(S)  *,20(I2,1H(,I1,2H) ))
        PRINT156,(WAITT(K),K=1,J)
156     FORMAT(5X,*WAITT  *,20F5.1)
        RETURN
        END
$IBFTC JPPEDJ
        SUBROUTINE SCANCL(STATE)
C
C     THIS SUB UPDATES THE COIL STATUS TO 1,2,3,0 OR 9
```

```
C       PUTTING SOME COIL IN PREFERRED WAIT D IF SO REQUIRED.
C       ALSO UPDATES CONTACTS OF DELAYS AT THE END IT DETERMINES
C       STATE (=1 FOR TRANSIENT, =2 FOR INIDT , =3 FOR STEADY STATE )
        INTEGER CN,VA,VB,COILJ,PREVST,P1,P2,P3,P4,SCANST,QENTRY,STATE
        INTEGER REQST(18)
        INTEGER V1(99),V2(99),TYPE(99),IDA(16),COILES(18),VACS(18),
     1  VBCS(18),VADDS(18),VBDDS(16),KOUKT(18),KTE(16,12),
     2  EJST(99),EXCASE(16),ADD(20),NEBAR(99,8),
     3  DIAGN,OPA(16),PRLEJ(99),QPED,QPSD,VALQ(99),
     4  EVNTEJ(16),EVNTST(16),KOUSC(18),PRIOCL(20),KOUSER(18),
     5  INAST(16),OPAST(16),PRST(18)
        DIMENSION PUD(18),DOD(18),WAITT(20)
        COMMON V1,V2,TYPE,IDA,COILES,VACS,VBCS,VADDS,VBDDS,KOUKT,KTE,
     1  EJST,EXCASE,ADD,NEBAR,DIAGN,PUD,DOD,KOUIN,KOUOP,KOUOD,KOUC,MAXEJ,
     2  MAXV,OPA,PRLEJ,QPED,QPSD,VALQ,EVNTEJ,EVNTST,KOUSC,NEVE,PRIOCL,
     3  KOUSER,WAITT,INAST,OPAST,PRST,JPOWER
        IF(JPOWER.NE.1) GOTO 501
        CALL STTOPA
        JPOWER=2
        STATE=1
        RETURN
501     CONTINUE
        DO599 CN=1,KOUC
        VA=VACS(CN)
        VB=VBCS(CN)
        JEXT=EXCASE(CN)
        GOTO(510,510,510,540),JEXT
510     CALL PATH(VA,01,P1)
        IF(P1.EQ.0) GOTO549
        CALL PATH(VB,02,P2)
        IF(P2.EQ.0) GOTO 549
        IF(EXCASE(CN).EQ.1) GOTO 548
        IF(EXCASE(CN).EQ.2) GOTO 521
        CALL PATH(VB,01,P4)
        IF(P4.EQ.1) GOTO 549
        GOTO 548
521     CALL PATH(VA,02,P3)
        IF(P3.EQ.1) GOTO 549
        GOTO 548
C
C
540      CALL PATH(VA,01,P1)
        CALL PATH(VB,02,P2)
        CALL PATH(VA,02,P3)
         CALL PATH(VB,01,P4)
        IF((P1*P2*(1-P3)*(1-P4).EQ.1).OR.
     1  ((1-P1)*(1-P2)*P3*P4.EQ.1)) GOTO 548
549     SCANST=0
        GOTO 550
548     SCANST=1
550     CONTINUE
        REQST(CN)=SCANST
599     CONTINUE
        KTCH=0
        DO699 CN=1,KOUC
C
        JP=PRST(CN)
        JS=REQST(CN)
```

```
      IF((JP.EQ.0).AND.(JS.EQ.0)) GOTO 699
      IF((JP.EQ.1).AND.(JS.EQ.1)) GOTO 699
      IF((JP.EQ.8).AND.(JS.EQ.1)) GOTO 699
      IF((JP.EQ.9).AND.(JS.EQ.0)) GOTO 699
C     ELSE COILSTATUS CHANGE OCCURED
      KOUSC(CN)=KOUSC(CN)+1
      IF(KOUSC(CN).LT.5) GOTO 615
      PRINT613,COILEJ
613   FORMAT(//20X,*CYCLING OCURRING THROUGH RELAY(EJ= *,
     1 I2,*).  PROCESSING TERMINATED*)
      STOP
615   IF((JP.EQ.0).AND.(JS.EQ.1)) GOTO 11
      IF((JP.EQ.2).AND.(JS.EQ.0)) GOTO 631
      IF((JP.EQ.1).AND.(JS.EQ.0)) GOTO 61
      IF((JP.EQ.3).AND.(JS.EQ.1)) GOTO 631
      IF((JP.EQ.2).AND.(JS.EQ.1)) GOTO 81
      IF((JP.EQ.3).AND.(JS.EQ.0)) GOTO 81
C   ELSE(JP=8,JS=0) OR (JP=9,JS=1)
      GOTO 71
C     JP=0 ' JS=1  (COIL IS TO BE PICKED-UP)
11    IF(PUD(CN).EQ.0.0) GOTO 621
      DELAY=PUD(CN)
      CALL ENTPRQ(CN,DELAY)
      PRST(CN)=8
C     THEN DONOT RETREAT
C     THEN DONOT RETREAT. CONTINUE SCAN
      GOTO 699
621   KTCH=1
      CALL UPQE(CN,1,QENTRY)
      IF(QENTRY.EQ.1) GOTO 622
      PRST(CN)=1
      GOTO 699
622   PRST(CN)=2
      GOTO 699
C     JP=2 JS=0
631   KTCH=1
      CALL KFS(CN,JS)
      PRST(CN)=JS
      GOTO 699
C     JP=1,JS=0 RELAY IS TO BE DRPPED
61    IF(DDD(CN).EQ.0.0) GOTO 641
      DELAY=DDD(CN)
      CALL ENTPRQ(CN,DELAY)
      PRST(CN)=9
      GOTO 699
641   KTCH=1
      CALL UPQE(CN,0,QENTRY)
      IF(QENTRY.EQ.1) GOTO 642
      PRST(CN)=0
      GOTO 699
642   PRST(CN)=3
      GOTO 699
71    CALL WITHDR(CN)
      PRST(CN)=JS
      GOTO 699
81    KTCH=1
      CALL KFS(CN,JS)
      PRST(CN)=JS
```

```
699     CONTINUE
        DO 1 CN=1,KOUC
        COILEJ=COILES(CN)
        EJST(COILEJ)=PRST(CN)
801     CONTINUE
        CALL STTOPA
C    THEN CHECK WHICH STATE IS TO BE REPORTED
        IF(KTCH.EQ.0) GOTO 821
        STATE=1
        RETURN
821     CONTINUE
        JCNQ=PRIOCL(1)
        IF(PRIOCL(1).EQ.0) GOTO 831
        STATE=2
        RETURN
831     STATE=3
        RETURN
        END
$IBFTC JPPEDK
        SUBROUTINE SERADQ(SLOCYC)
C
C       THIS SUB. SERVES THE WAIT-Q FOR LOWEST WAITT COILS  AND
C       ADJUST Q
C       INFO ABOUT THE TIMERS WHICH TIMED-OUT
C
        INTEGER EJ
        INTEGER CN,COILEJ,COILST,PEJ(20),QENTRY,SLOCYC
        INTEGER V1(99),V2(99),TYPE(99),INA(16),COILES(18),VACS(18),
     1  VBCS(18),VAODS(16),VBODS(16),KOUKT(18),KTE(18,12),
     2  EJST(99),EXCASE(18),ADD(20),NEBAR(99,8),
     3  DIAGN,OPA(16),PRLEJ(99),QPED,QPSD,VALQ(30),
     4  EVNTEJ(16),EVNTST(16),KOUSC(18),PRIOCL(20),KOUSER(18),
     5  INAST(16),OPAST(16),PRST(18)
        DIMENSION PUD(18),DOD(18),WAITT(20)
        COMMON V1,V2,TYPE,INA,COILES,VACS,VBCS,VAODS,VBODS,KOUKT,KTE,
     1  EJST,EXCASE,ADD,NEBAR,DIAGN,PUD,DOD,KOUIN,KOUOP,KOUOD,KOUC,MAXEJ,
     2  MAXV,OPA,PRLEJ,QPED,QPSD,VALQ,EVNTEJ,EVNTST,KOUSC,NEVE,PRIOCL,
     3  KOUSER,WAITT,INAST,OPAST,PRST,JPOWER
C
        N=0
        SWT=WAITT(1)
1       CN=PRIOCL(1)
        KOUSER(CN)=KOUSER(CN)+1
        IF(KOUSER(CN).GT.5) GOTO 451
        COILEJ=COILES(CN)
        COILST=EJST(COILEJ)
        N=N+1
        PEJ(N)=COILEJ
        IF(COILST.EQ.8) GOTO 31
C       ELSE COILST IS 9 THIS COIL SHOULD DROP OUT NOW
        CALL UPQE(CN,0,QENTRY)
        IF(QENTRY.EQ.1) GOTO 34
        EJST(COILEJ)=0
        PRST(CN)=0
        GOTO 50
34      EJST(COILEJ)=8
        PRST(CN)=3
        GOTO 50
```

```
C       THIS RELAY TO PICK-UP NEW
31      CALL UPQE(CN,1,Q NTRY)
        IF(QENTRY.EQ.1) GOTO 44
        EJST(COILEJ)=1
        PRST(CN)=1
        GOTO50
44      EJST(COILEJ)=2
        PRST(CN)=2
50      CONTINUE
C      NOW ADJUST THE Q BY SHIFTING THE COILS TO LEFT ONE POSITION TILL
C        THE SHIFTED COIL IS A
        K=1
71      PRIOCL(K)=PRIOCL(K+1)
        WAITT(K)=WAITT(K+1)
        IF(PRIOCL(K).EQ.0) GOTO 87
        K=K+1
        GOTO 71
C      NOW SEE IF THE COIL IN THE FIRST PLACE OF THE ADJUSTED Q
C       SHOULD ALSO BE SERVICED.
87      IF(WAITT(1).EQ.SWT) GOTO 1
C       REDUCE THE WAITT OF THE COILS BY SWT
        DO356 J=1,20
        IF(PRIOCL(J).EQ.0) GOTO 391
        WAITT(J)=WAITT(J)-SWT
356     CONTINUE
391     SLOCYC=0
        PRINT399,(PEJ(L),L=1,N)
399     FORMAT(//2X,2H*),* RELAY TIME-OUT(EJ=) *,20(I2,1X))
        JPOWER=1
C     THIS ENSURES EFFECT OF THESE CONTACT UPDATINGS ON OP IS NOT LOST
        RETURN
451     PRINT455,COILEJ
455     FORMAT(//15X,*SLOW CYCLING THROUGH TCR(EDGE)= *,
      1 I2,* OCCURING. PROCESSING TERMINATED*)
        SLOCYC=1
        RETURN
        END
$IBFTC JPPEDM
        SUBROUTINE STTOPA
C
C.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-
C       THIS SUB. PREPARES THE OPAST ARRAY FOR OUTPUT REPORT
        INTEGER OPN,VA,VB,P1,P2,EJ
        INTEGER V1(99),V2(99),TYPE(99),INA(16),COILES(18),VACS(18),
      1 VBCS(18),VAODS(16),VBODS(16),KOUKT(18),KTE(18,12),
      2 EJST(99),EXCASE(18),ADD(20),NEBAR(99,8),
      3 DIAGN,OPA(16),PRLEJ(99),QPED,QPSD,VALQ(30),
      4 EVNTEJ(16),EVNTST(16),KOUSC(18),PRIOCL(20),KOUSER(18),
      5 INAST(16),OPAST(16),PRST(18)
        DIMENSION PUD(13),DOD(18),WAITT(20)
        COMMON V1,V2,TYPE,INA,COILES,VACS,VBCS,VAODS,VBODS,KOUKT,KTE,
      1 EJST,EXCASE,ADD,NEBAR,DIAGN,PUD,DOD,KOUIN,KOUOP,KOUOD,KOUC,MAXEJ,
      2 MAXV,OPA,PRLEJ,QPED,QPSD,VALQ,EVNTEJ,EVNTST,KOUSC,NEVE,PRIOCL,
      3 KOUSER,WAITT,INAST,OPAST,PRST,JPOWER
C        FIRST SETTLE OD,S
        DO19 OPN=1,KOUOD
        VA=VAODS(OPN)
        VB=VBODS(OPN)
```

```
      CALL PATH(VA,V1,P1)
      IF(P1.EQ.0) GOTO 11
      CALL PATH(VB,V2,P2)
      IF(P2.EQ.0) GOTO 11
C     ELSE THIS OP IS ON
      OPAST(OPN)=1
      GOTO 45
11    OPAST(OPN)=0
45    EJ=OPA(OPN)
      EJST(EJ)=OPAST(OPN)
49    CONTINUE
C     THEN SETTLE ADDITIONAL OP'S IF ANY
      IF(KOUOP.EQ.KOUOD) GOTO 99
      NEXT=KOUOD+1
      DO79 OPN=NEXT,KOUOP
      EJ=OPA(OPN)
      OPAST(OPN)=EJST(EJ)
79    CONTINUE
99    RETURN
      END
$IBFTC JPPEDN
      SUBROUTINE UPISTA
C
C     THIS SUB IMPLANTS THE INPUT EVENT INTO THE SYSTEM AND
C      PREPARES CBE INAST ARRAY FOR OUTPUT REPORT
C
      INTEGER EJUS
      INTEGER V1(99),V2(99),TYPE(99),INA(16),COILES(18),VACS(18),
     1 VBCS(18),VAODS(16),VBODS(16),KOUKT(18),KTE(18,12),
     2 EJST(99),EXCASE(18),ADD(20),NEBAR(99,8),
     3 DIAGN,OPA(16),PRLEJ(99),QPED,QPSD,VALQ(30),
     4 EVNTEJ(16),EVNTST(16),KOUSC(18),PRIOCL(20),KOUSER(18),
     5 INAST(16),OPAST(16),PRST(18)
      DIMENSION PUD(18),DOD(18),WAITT(20)
      COMMON V1,V2,TYPE,INA,COILES,VACS,VBCS,VAODS,VBODS,KOUKT,KTE,
     1 EJST,EXCASE,ADD,NEBAR,DIAGN,PUD,DOD,KOUIN,KOUOP,KOUOD,KOUC,MAXEJ,
     2 MAXV,OPA,PRLEJ,QPED,QPSD,VALQ,EVNTEJ,EVNTST,KOUSC,NEVE,PRIOCL,
     3 KOUSER,WAITT,INAST,OPAST,PRST,JPOWER
C      FIRST FIND THE COUNT KE OF INPUT CHANGES
      DO11 J=1,16
      IF(EVNTEJ(J).EQ.0) GOTO 21
11    CONTINUE
      KE=16
      GOTO 31
21    KE=J-1
C     THEN DETERMINE THE INA NUMBER AND IF FOUND
C      IMPLANT STATUS IN EJST & INAST
C
31    DO59 J=1,KE
      EJUS=EVNTEJ(J)
      DO49 K=1,KOUIN
      IF(INA(K).NE.EJUS) GOTO 45
C     ELSE K IS THE INA NUMBER FOR THIS INPUT EDGE
      EJST(EJUS)=EVNTST(J)
      INAST(K)=EVNTST(J)
      GOTO 59
45    IF(K.EQ.KOUIN) GOTO 61
49    CONTINUE
```

```
59      CONTINUE
        RETURN
81      PRINT85,NEVE,EJJS
85      FORMAT(//15X,*REFER EVENT NO. *,I2,
     1 *). EJ=*,I2,*IS NOT AN INPUT EJ.  PROCESSING TERMINATED.*)
        STOP
        END
$IBFTC JPP.DP
        SUBROUTINE UPQE(CN,ST,QENTRY)
C
C
C    THIS SUBROUTINE UPDATESCONTACTS ON A CHNGE OF COIL STATUS
C    AND PUTS THE COIL IN TRANSIENT,S Q IF ANY CNTACTS ARE PUT IN
C     TRANSIENT STATE
C
        INTEGER QENTRY,EJ,ST,CN
        INTEGER V1(99),V2(99),TYPE(99),INA(16),COILES(18),VACS(18),
     1 VBCS(18),VAODS(16),VBODS(16),KOUKT(18),KTE(18,12),
     2 EJST(99),EXCASE(18),ADD(20),NEBAR(99,8),
     3 DIAGN,OPA(16),PRLEJ(99),QPED,QPSD,VALQ(30),
     4 EVNTEJ(16),EVNTST(16),KOUSC(18),PRIOCL(20),KOUSER(18),
     5 INAST(16),OPAST(16),PRST(18)
        DIMENSION PUD(19),DOD(18),WAITT(20)
        COMMON V1,V2,TYPE,INA,COILES,VACS,VBCS,VAODS,VBODS,KOUKT,KTE,
     1 EJST,EXCASE,ADD,NEBAR,DIAGN,PUD,DOD,KOUIN,KOUOP,KOUOD,KOUC,MAXEJ,
     2 MAXV,OPA,PRLEJ,QPED,QPSD,VALQ,EVNTEJ,EVNTST,KOUSC,NEVE,PRIOCL,
     3 KOUSER,WAITT,INAST,OPAST,PRST,JPOWER
C
        QENTRY=0
        MAXKT=KOUKT(CN)
        DO99 KTN=1,MAXKT
        EJ=KTE(CN,KTN)
        JT=TYPE(EJ)
        IF(((ST.EQ.1).AND.((JT.EQ.2).OR.(JT.EQ.3).OR.(JT.EQ.4))).OR.
     1 ((ST.EQ.0).AND.((JT.EQ.1).OR.(JT.EQ.3).OR.(JT.EQ.4)))) GOTO 1
        EJST(EJ)=1
        GOTO 2
1       EJST(EJ)=0
C     CHECK IF ONE OF THE TRANSFER CONTACTS HAS BEEN FOUND
C
2       IF(JT.LE.2) GOTO 99
        QENTRY=1
99      CONTINUE
        RETURN
        END
$IBFTC JPP.DQ
        SUBROUTINE WITHDR(CN)
C
C       THIS SUB. REMOVES COIL CN FROM THE WAIT-Q AND READJUSTS THE Q
        INTEGER CN
        INTEGER V1(99),V2(99),TYPE(99),INA(16),COILES(18),VACS(18),
     1 VBCS(18),VAODS(16),VBODS(16),KOUKT(18),KTE(18,12),
     2 EJST(99),EXCASE(18),ADD(20),NEBAR(99,8),
     3 DIAGN,OPA(16),PRLEJ(99),QPED,QPSD,VALQ(30),
     4 EVNTEJ(16),EVNTST(16),KOUSC(18),PRIOCL(20),KOUSER(18),
     5 INAST(16),OPAST(16),PRST(18)
        DIMENSION PUD(19),DOD(18),WAITT(20)
        COMMON V1,V2,TYPE,INA,COILES,VACS,VBCS,VAODS,VBODS,KOUKT,KTE,
```

```
      1 EJ  ,XXCASE,ADD, R,  ,  ,PUD,DOD,KOUIN,KOUDP,KOUOD,KOUC,MAXEJ,
      2 MAXV,OPA,PRLEJ,QP  D,QFSE,VALU, VNT-J,EVNTST,KOUSC,NEVE,PRIOCL,
      3 KOUSLR,WAITT,INAST, PAST,PRST,JPOWER
C
C       IT IS CERTAIN THAT WE WILL LOCATE CN BEFORE OR AT J=18 HENCE
C          VALUE OF J WILL BE AVAILABLE OUTSIDE DO LOOP DUE TO
C           ABNORMAL EXIT
        DO3 J=1,20
        IF(PRIOCL(J).EQ.CN) GOTO 11
3       CONTINUE
11      CONTINUE
C       CN IS SITTING AT POSITION J REMOVE IT FROM THERE ' ADJUST Q
        DO15 K=J,19
        PRIOCL(K)=PRIOC_(K+1)
        WAITT(K)=WAITT(K+1)
        IF(PRIOCL(K).EQ. ) GOTO 99
15      CONTINUE
99      CONTINUE
        RETURN
        END
```

CSP-1979-M-DIX-COM